МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Операционные системы»
Тема: Обработка стандартных прерываний

Студент гр. 9382 _____ Балаева М.О.

Преподаватель _____ Ефремов М.А

Санкт-Петербург
2021

Цель работы: В архитектуре компьютера существуют стандартные прерывания, за которыми закреплены определённые вектора прерываний. Вектор прерываний хранит адрес подпрограммы обработчика прерываний. При возникновении прерывания, аппаратура компьютера передаёт управление по соответствующему адресу вектора прерывания. Обработчик прерываний получает управление и выполняет соответствующие действия.

В лабораторной работе № 4 предлагается построить обработчик прерываний сигналов таймера. Эти сигналы генерируются аппаратурой через определённые интервалы времени и, при возникновении такого сигнала, возникает прерывание с определённым значением вектора. Таким образом, управление будет передано функции, чья точка входа записана в соответствующий вектор прерывания.

Описание функций.

Название функции	**Назначение**
ROUT	пользовательский обработчик прерываний, считающий и печатающий количество его вызовов
OUT_BP	функция вывода строки по адресу ES:BP
PRINT	вызывает функцию печати строки
PROV_ROUT	Проверка на установленный пользовательский обработчик с последующей его установкой. В случае, если хвост равен ‘/un’, восстанавливает стандартный
SET_ROUT	устанавливает пользовательское прерывание
DEL_ROUT	удаляет пользовательское прерывание
SAVE_STAND	сохраняет адрес стандартного прерывания в KEEP_IP, KEEP_CS
BYTE_TO_HEX	переводит число AL в коды

	символов 16-ой с/с, записывая получившееся в al и ah
TETR_TO_HEX	вспомогательная функция для работы функции BYTE_TO_HEX
WRD_TO_HEX	переводит число AX в строку в 16-ой с/с, записывая получившееся в di, начиная с младшей цифры

Последовательность действий, выполняемых утилитой.

1) Проверка состояния памяти до запуска утилиты, при помощи lab3_1.com (рис. 1).

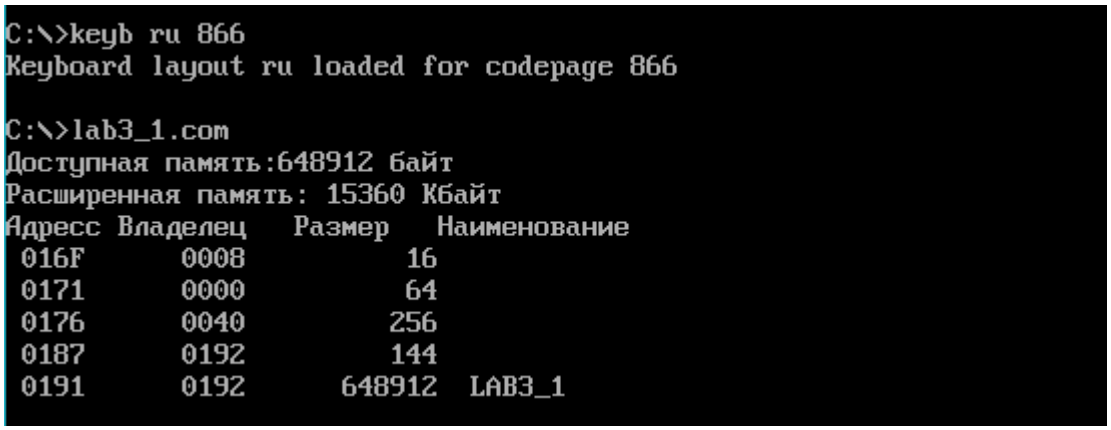

Рис. 1

2) Загрузка утилиты lab4 в память и очередной вывод состояния памяти (рис. 2).

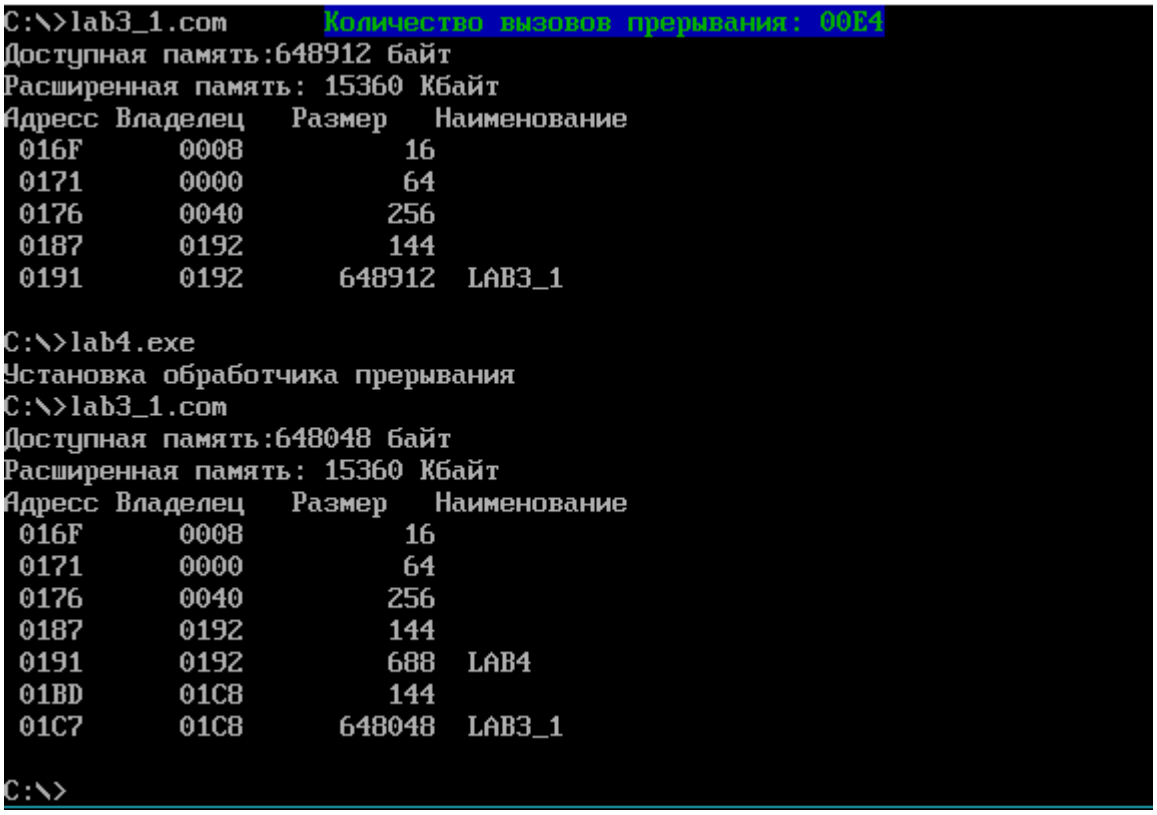

Рис. 2

3) Повторный запуск lab4 (рис. 3).

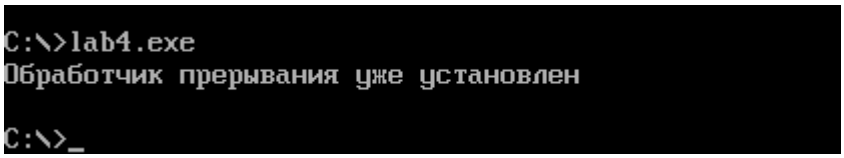

Рис. 3

4) Запустим программу lab4.exe с ключом выгрузки /un и последующий вызов lab3_1.com (рис. 4).

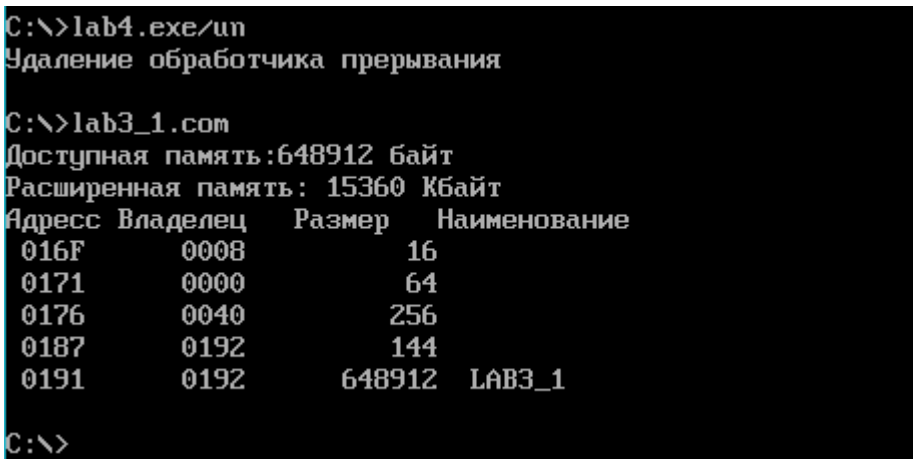

Рис. 4

Заключение.

В процессе выполнения данной лабораторной работы был построен обработчик прерываний сигналов таймера.

Ответы на контрольные вопросы.

- Как реализован механизм прерывания от часов?

Сначала сохраняется содержимое регистров, потом определяется источник прерывания, по номеру которого определяется смещение в таблице векторов прерывания, сохраняется в CS : IP, передаётся управление по адресу CS:IP и происходит выполнение обработчика, и в конце происходит возврат управления прерванной программе. Прерывания генерируются системным таймером с частотой 18,206 Гц.

- Какого типа прерывания использовались в работе?

В программе использовалось аппаратное прерывание от системного таймера.

ПРИЛОЖЕНИЕ А

lab4.asm

```
CODE SEGMENT
 ASSUME CS:CODE, DS:DATA, ES:DATA, SS:STACK

ROUT PROC FAR
      jmp mark
      SGNTR dw 0ABCDh
      KEEP_PSP dw 0
      KEEP_IP dw 0 ;
      KEEP_CS dw 0
      COUNT dw 0
      NUM_CALL db 'Количество вызовов прерывания:     $'
      KEEP_AX dw 0
      KEEP_SS dw 0
      KEEP_SP dw 0
      INT_STACK dw 100 dup (?)
      mark:
      mov KEEP_SS, SS
      mov KEEP_SP, SP
      mov KEEP_AX, AX
      mov AX,seg INT_STACK
      mov SS,AX
      mov SP,0
      mov AX,KEEP_AX


      push ax
      push bp
      push es
      push ds
      push dx
      push di

      mov ax,cs
      mov ds,ax
      mov es,ax
      mov ax,CS:COUNT
      add ax,1
      mov CS:COUNT,ax
      mov di,offset num_call+34
      call WRD_TO_HEX
      mov bp,offset num_call
      call OUT_BP

      pop di
      pop dx
      pop ds
      pop es
      pop bp
      mov al,20h
      out 20h,al
      pop ax
      mov   AX,KEEP_SS
      mov   SS,AX
      mov   AX,KEEP_AX
      mov   SP,KEEP_SP
      iret
ROUT ENDP; ------------------------------------
TETR_TO_HEX PROC near
```

```
      and AL,0Fh
      cmp AL,09
      jbe NEXT
      add AL,07
NEXT: add AL,30h
      ret
TETR_TO_HEX ENDP
;-------------------------------------
BYTE_TO_HEX PROC near
      push CX
      mov AH,AL
      call TETR_TO_HEX
      xchg AL,AH
      mov CL,4
      shr AL,CL
      call TETR_TO_HEX
      pop CX
      ret
BYTE_TO_HEX ENDP
;-------------------------------------

WRD_TO_HEX PROC near
      push BX
      mov BH,AH
      call BYTE_TO_HEX
      mov [DI],AH
      dec DI
      mov [DI],AL
      dec DI
      mov AL,BH
      call BYTE_TO_HEX
      mov [DI],AH
      dec DI
      mov [DI],AL
      pop BX
      ret
WRD_TO_HEX ENDP
;-------------------------------------
;
OUT_BP PROC near
      push ax
      push bx
      push dx
      push cx
      mov ah,13h
      mov al,0
      mov bl,12h
      mov bh,0
      mov dh,0
      mov dl,20
      mov cx,35
      int 10h
      pop cx
      pop dx
      pop bx
      pop ax
      ret
OUT_BP ENDP
LAST_BYTE:
;-------------------------------------
PRINT PROC
```

```
      push ax
      mov ah,09h
      int 21h
      pop ax
      ret
PRINT ENDP
;--------------------------------------

PROV_ROUT PROC
      mov ah,35h
      mov al,1ch
      int 21h
      mov si,offset SGNTR
      sub si,offset ROUT
      mov ax,0ABCDh
      cmp ax,ES:[BX+SI]
      je ROUT_EST
            call SET_ROUT
            jmp PROV_END
      ROUT_EST:
            call DEL_ROUT
      PROV_END:
      ret
PROV_ROUT ENDP
;--------------------------------------

SET_ROUT PROC
      mov ax,KEEP_PSP
      mov es,ax ;
      cmp byte ptr es:[80h],0
            je SH
      cmp byte ptr es:[82h],'/'
            jne SH
      cmp byte ptr es:[83h],'u'
            jne SH
      cmp byte ptr es:[84h],'n'
            jne SH

      mov dx,offset DONT_SET
      call PRINT
      ret

      SH:
      ; сохраняем стандартный обработчик:
      call SAVE_HAND

      mov dx,offset SET
      call PRINT

      push ds
      ; кладём в ds:dx адрес нашего обработчика:
      mov dx,offset ROUT
      mov ax,seg ROUT
      mov ds,ax

      ; меняем адрес обработчика прерывания 1Ch:
      mov ah,25h
      mov al,1ch
      int 21h
      pop ds
```

```
      ; оставляем программу резидентно:
      mov dx,offset LAST_BYTE
      mov cl,4
      shr dx,cl ; делим dx на 16
      add dx,1
      add dx,20h

      xor AL,AL
      mov ah,31h
      int 21h ; оставляем наш обработчик в памяти

      xor AL,AL
      mov AH,4Ch
      int 21H
SET_ROUT ENDP
;--------------------------------------
; удаление нашего обработчика:
DEL_ROUT PROC
      push dx
      push ax
      push ds
      push es


      mov ax,KEEP_PSP
      mov es,ax ; кладём в es PSP нашей програмы
      cmp byte ptr es:[80h],0
            je DELL_END
      cmp byte ptr es:[82h],'/'
            jne DELL_END
      cmp byte ptr es:[83h],'u'
            jne DELL_END
      cmp byte ptr es:[84h],'n'
            jne DELL_END

      mov dx,offset DELL
      call PRINT

      mov ah,35h
      mov al,1ch
      int 21h
      mov si,offset KEEP_IP
      sub si,offset ROUT


      mov dx,es:[bx+si]
      mov ax,es:[bx+si+2]
      mov ds,ax
      mov ah,25h
      mov al,1ch
      int 21h


      mov ax,es:[bx+si-2]
      mov es,ax
      mov ax,es:[2ch]
      push es
      mov es,ax
      mov ah,49h
      int 21h
      pop es
```

```
      mov ah,49h
      int 21h

      jmp DELL_END2

      DELL_END:
      mov dx,offset YET_SET
      call PRINT
      DELL_END2:

      pop es
      pop ds
      pop ax
      pop dx
      ret
DEL_ROUT ENDP


SAVE_HAND PROC
      push ax
      push bx
      push es
      mov ah,35h
      mov al,1ch
      int 21h
      mov KEEP_CS, ES
      mov KEEP_IP, BX
      pop es
      pop bx
      pop ax
      ret
SAVE_HAND ENDP
;--------------------------------------
BEGIN:
      mov ax,DATA
      mov ds,ax
      mov KEEP_PSP, es
      call PROV_ROUT
      xor AL,AL
      mov AH,4Ch
      int 21H
CODE ENDS

STACK SEGMENT STACK
      dw 100h dup (?)
STACK ENDS

DATA SEGMENT
      SET db 'Установка обработчика прерывания','$'
      DELL db 'Удаление обработчика прерывания',0DH,0AH,'$'
      YET_SET db 'Обработчик прерывания уже установлен',0DH,0AH,'$'
      DONT_SET db 'Обработчик прерывания не установлен',0DH,0AH,'$'
      STRENDL db 0DH,0AH,'$'
DATA ENDS

 END BEGIN
```