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

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Операционные системы»
Тема: Сопряжение стандартного и пользовательского обработчиков прерываний

Студентка гр. 9382 _____ Балаева М.О.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Исследование возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры. Пользовательский обработчик прерывания получает управление по прерыванию (int 09h) при нажатии клавиши на клавиатуре. Он обрабатывает скан-код и осуществляет определенные действия, если скан-код совпадает с определенными кодами, которые он должен обрабатывать. Если скан-код не совпадает с этими кодами, то управление передается стандартному прерыванию.

Описание функций и структур данных.

Название функции	**Описание функции**
ROUT	Обработчик прерывания, который интерпретирует нажатие на клавишу ‘t’ как символ ‘$’
PRINT	Печать строки
PROV_ROUT	Проверка на установленный обработчик, при использовании ключа удаляет его
SET_ROUT	Устанавливает пользовательское прерывание
DEL_ROUT	Удаляет пользовательское прерывание
SAVE_HAND	Сохраняет адрес стандартного прерывания

Ход работы и функционал программы.

- Проверка состояния памяти до запуска lab5.exe с помощью утилиты из предыдущих работах (lab3_1.com) на рис.1:

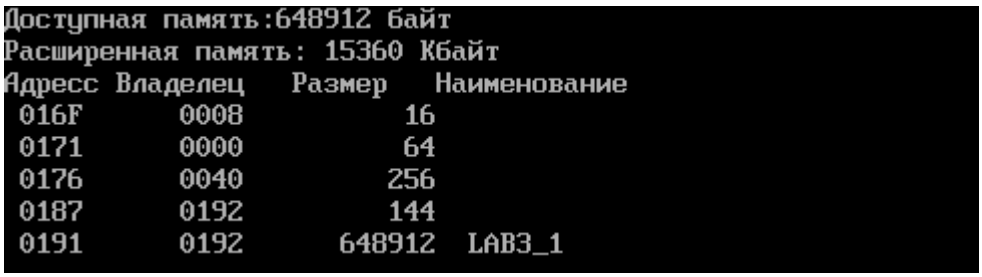

Рис.1

- Запуск программы Lab5.exe с последующей проверкой состояния памяти, а также примером работы программы на рис.2:

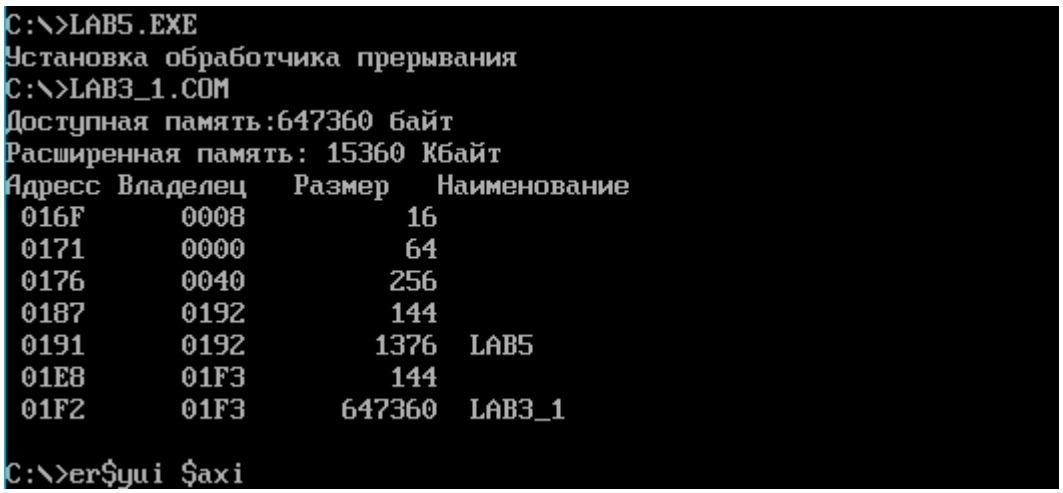

Рис.2

- Повторный запуск программы на рис.3:

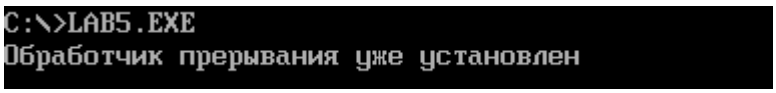

Рис.3

- Запуск Lab5.exe с ключом выгрузки и последующей проверкой состояния памяти на рис.4:

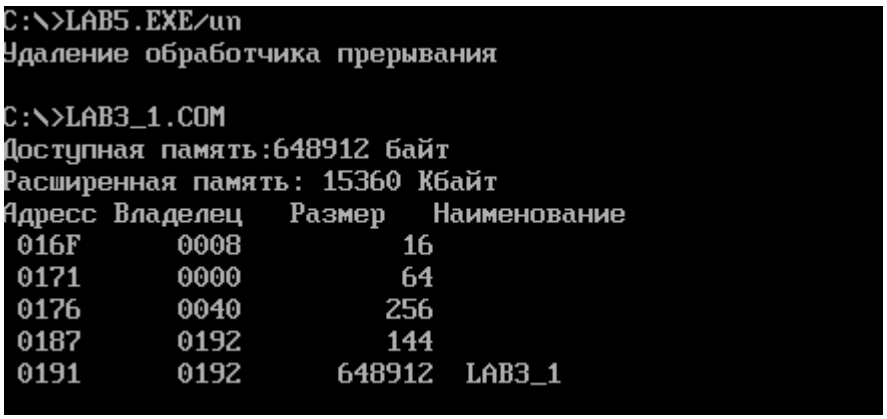

Рис.4

Заключение.

В процессе выполнения данной лабораторной работы была исследована возможность встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры.

Ответы на контрольные вопросы.

1) *Какого типа прерывания использовались в работе?*

   В работе использовались прерывания BIOS 9h,16h и пользовательское прерывание DOS 21h.

2) *Чем отличается скан код от кода ASCII?*

   Скан код – это код нажатой клавиши, используемый для распознавания драйвером клавиатуры, а код ASCII – это код символа, закрепленного за клавишей

ПРИЛОЖЕНИЕ А

lab5.asm

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

ROUT PROC FAR

      jmp mark
      SGNTR dw 0ABCDh
      KEEP_PSP dw 0
      KEEP_IP dw 0
      KEEP_CS dw 0
      COUNT dw 0
      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 es
      push ds
      push dx
      push di
      push cx
      mov al,0
      in al,60h
      cmp al,14h
      je mark1

      pushf
      call dword ptr cs:KEEP_IP
      jmp quite

      mark1:
      in al, 61h
      mov ah, al
      or al, 80h
      out 61h, al
      xchg ah, al
      out 61h, al
      mov al, 20h
      out 20h, al

      buf_push:
      mov al,0
      mov ah,05h ; код функции
      mov cl,24h ; код символа
      mov ch,00h
      int 16h
      or al,al
      jz quite
      mov ax,0040h
```

```
      mov es,ax
      mov ax,es:[1Ah]
      mov es:[09h],ax
      jmp buf_push

      quite:
      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
LAST_BYTE:
;--------------------------------------
PRINT PROC
      push ax
      mov ah,09h
      int 21h
      pop ax
      ret
PRINT ENDP
;--------------------------------------
PROV_ROUT PROC
      mov ah,35h
      mov al,09h
      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:
      call SAVE_HAND

      mov dx,offset SET
      call PRINT

      push ds
      mov dx,offset ROUT
      mov ax,seg ROUT
      mov ds,ax

      mov ah,25h
      mov al,09h
      int 21h
      pop ds


      mov dx,offset LAST_BYTE
      mov cl,4
      shr dx,cl
      add dx,1
      add dx,40h

      xor AL,AL
      mov ah,31h
      int 21h

      ret
SET_ROUT ENDP
;--------------------------------------

DEL_ROUT PROC
      push dx
      push ax
      push ds
      push es


      mov ax,KEEP_PSP
      mov es,ax
      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

      CLI

      mov ah,35h
      mov al,09h
      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,09h
      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

      STI
      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,09h
      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

ASTACK SEGMENT STACK
      dw 100h dup (?)
ASTACK 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
```