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.

Студент гр. 9381 Шахин Н.С

Преподаватель Ефремов М. А.

Санкт-Петербург

2021

Цель работы.

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

Ход работы.

1) Написан и отлажен программный модуль типа .EXE, который выполняет такие же функции, как и в программе лабораторной работы №4, а именно:

- Проверяет, установлено ли пользовательское прерывание с вектором 09h.
- Если прерывание не установлено, то устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний. Адрес точки входа в стандартный обработчик прерывания находится в теле пользовательского обработчика. Осуществляется выход по функции 4Ch прерывания int21h.
- Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h.

2) Запустил отлаженную программу и убедился, что резидентный обработчик прерывания 09h установлен. Работа прерывания была проверена введением различных символов, обрабатываемых установленным обработчиком и стандартным обработчиком.

```
F:\>lab5.exe
Download custom interrupt successfully completed.
The list of the traced keyboard shortcuts:
 - (Ctrl+Alt+H): Output of this help.
 - (Ctrl+Alt+C): Output of the call counter of the processor.
```

Управление передается стандартному обработчику.

Ввел комбинации клавиш Ctrl+Alt+H и Ctrl+Alt+C.

```
F:\>lab5.exe
Download custom interrupt successfully completed.
The list of the traced keyboard shortcuts:
 - (Ctrl+Alt+H): Output of this help.
 - (Ctrl+Alt+C): Output of the call counter of the processor.
Counter of number of calls of the user interruption:    94 .
C
F:\>_
```

3) Проверил размещение прерывания в памяти. Для этого запустил программу лабораторной работы №3, которая отображает карту памяти в виде списка блоков MCB.

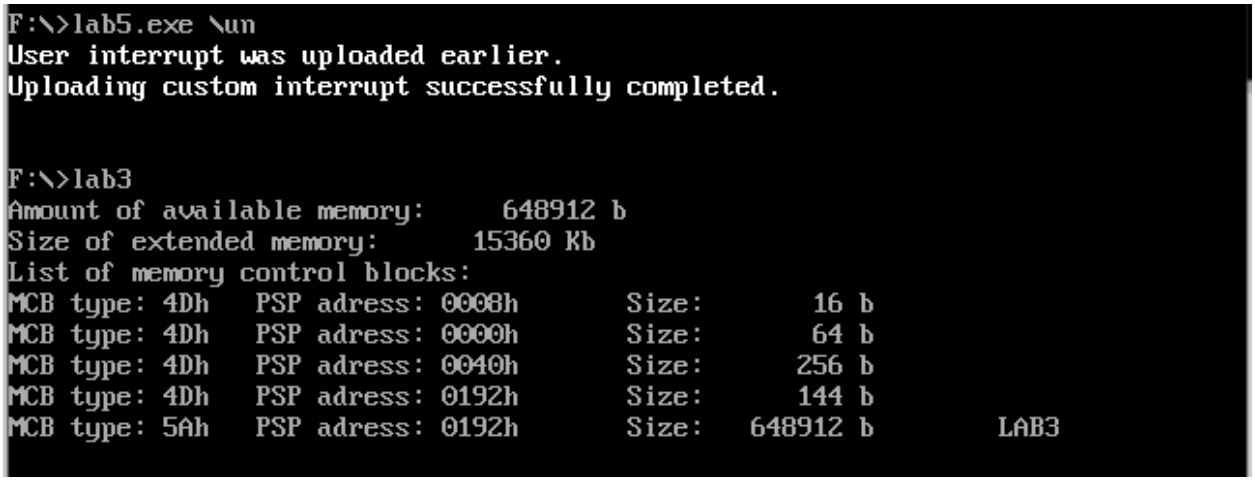
```
F:\>lab3
Amount of available memory:     646480 b
Size of extended memory:      15360 Kb
List of memory control blocks:
MCB type: 4Dh   PSP adress: 0008h       Size:       16 b
MCB type: 4Dh   PSP adress: 0000h       Size:       64 b
MCB type: 4Dh   PSP adress: 0040h       Size:      256 b
MCB type: 4Dh   PSP adress: 0192h       Size:      144 b
MCB type: 4Dh   PSP adress: 0192h       Size:     2256 b        LAB5
MCB type: 4Dh   PSP adress: 022Ah       Size:     2144 b
MCB type: 5Ah   PSP adress: 022Ah       Size:   646480 b        LAB3

F:\>_
```

Как видно, резидент находится в памяти и успешно используется.

4) Запустил отлаженную программу еще раз и убедился, что программа определяет установленный обработчик прерываний.

```
F:\>lab5.exe
User interrupt was uploaded earlier.
The list of the traced keyboard shortcuts:
 - (Ctrl+Alt+H): Output of this help.
 - (Ctrl+Alt+C): Output of the call counter of the processor.
```

5) Запустил отлаженную программу с ключом выгрузки и убедился, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память, занятая резидентом освобождена. Для этого также запустил программу лабораторной работы №3.



Резидент был успешно выгружен из памяти. При нажатии наших “горячих клавиш”, сообщения не выводятся.

Сведения о функциях и структурах данных управляющей программы:

Названия функций	Описание
INT_09H_PRO	Обработчик пользовательского прерывания
PR_STR_BIOS	Функция вывода текста
PR_CHR_BIOS	Функция вывода символа
DWRD_TO_DEC	Функция перевода BX:AX в десятичную с.с
WRD_TO_DEC	Функция перевода слова из AX в DEC
BYTE_TO_DEC	Функция перевода байта из AL в DEC
WRD_TO_HEX	Функция перевода в HEX слова из AX
BYTE_TO_HEX	Функция перевода байта из AL в два символа HEX
TETR_TO_HEX	Функция перевода десятичной цифры в код символа

Название переменных	Описание
PSP_SIZ	Хранит размер PSP
STK_SIZ	Хранит размер стека
KEEP_IP (dw)	Переменная для хранения смещения прерывания
KEEP_IP (dw)	Переменная для хранения сегмента прерывания
IT_CNTR (dw)	Счетчик
L5_SIGN (db)	Хранит текст: String = signature.

CMD_ERR (db)	Хранит текст: Error! At the end of the command line detected an invalid parameter.
UNL_ERR (db)	Хранит текст: Error!The program started with the key "/un" before the implementation of interrupts.
STA_LOA (db)	Хранит текст: Download custom interrupt successfully completed.
STA_ALR (db)	Хранит текст: User interrupt was uploaded earlier.
STA_UNL (db)	Хранит текст: Uploading custom interrupt successfully completed.
INF_USG (db)	Хранит текст: The list of the traced keyboard shortcuts:
INF_HK1 (db)	Хранит текст: - (Ctrl+Alt+H): Output of this help.
INF_HK2 (db)	Хранит текст: - (Ctrl+Alt+C): Output of the call counter of the processor.Interuption is loading now!
INT_CNT (db)	Хранит текст: Counter of number of calls of the user interruption:
PRM_ERR (db)	Хранит текст: Failed to release the memory used by the program. Error code: H.
ENM_ERR (db)	Хранит текст: Unable to free the memory occupied by the environment. Error code: H.

Вывод.

В ходе данной работы производилось исследование возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры. Было написано пользовательское прерывание от клавиатуры, которое анализирует скан-коды, выполняет вывод сообщения результата нажатия, а при несовпадении скан-кода передает управление стандартному обработчику.

Ответы на контрольные вопросы:

1. Какого типа прерывания использовались в работе?

Использовались следующие типы прерываний:

Аппаратные (прерывание от клавиатуры 09h)

Программные (прерывания вызываемые командой int 21h)

2. Чем отличается скан код от кода ASCII?

Скан-код – это код, который клавиатура передаёт системе. Тем самым система определяет, какая клавиша (или комбинация клавиш) была нажата. ASCII-код – это таблица кодировок для печатных символов.

Таким образом скан-код – это номер клавиши, а ASCII-код – код, соответствующий обозначению на этой клавише

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
.SEQ

L5_CODE SEGMENT
        ASSUME CS: L5_CODE, DS: L5_DATA, ES: NOTHING, SS: L5_STACK
START:  jmp l5_start

L5_DATA SEGMENT
        PSP_SIZ = 10h
        STK_SIZ = 10h
        CMD_BUF db 80h dup (00h)

        HK_KEY1 = 23h
        HK_ASC1 = 'H'
        HK_KEY2 = 2Eh
        HK_ASC2 = 'C'

        L5_SIGN db 'String = signature.',
                  0Dh, 0Ah, '$'
        CMD_ERR db 'Error! At the end of the command line detected an invalid
parameter.',                    0Dh, 0Ah, '$'
        UNL_ERR db 'Error!The program started with the key "/un" before the
implementation of interrupts.',   0Dh, 0Ah, '$'
        STA_LOA db 'Download custom interrupt successfully completed.',
                        0Dh, 0Ah, '$'
        STA_ALR db 'User interrupt was uploaded earlier.',
                        0Dh, 0Ah, '$'
        STA_UNL db 'Uploading custom interrupt successfully completed.',
                        0Dh, 0Ah, '$'
        INF_USG db 'The list of the traced keyboard shortcuts:',
                        0Dh, 0Ah, '$'
        INF_HK1 db ' - (Ctrl+Alt+H): Output of this help.',
                        0Dh, 0Ah, '$'
        INF_HK2  db  '  -  (Ctrl+Alt+C):  Output  of  the  call  counter  of  the
processor.',                          0Dh, 0Ah, '$'
        INT_CNT db 'Counter of number of calls of the user interruption:     .',
                          0Dh, 0Ah, '$'
        PRM_ERR db 'Failed to release the memory used by the program. Error
code:     H.',                0Dh, 0Ah, '$'
        ENM_ERR db 'Unable to free the memory occupied by the environment. Error
code:    H.',           0Dh, 0Ah, '$'

        KEEP_IP dw ?
        KEEP_CS dw ?
        IT_CNTR dw 0

        CHR_EOT = '$'
        INF_CLR = 0Fh

L5_DATA ENDS

L5_STACK SEGMENT STACK
        db STK_SIZ * 10h dup (?)
L5_STACK ENDS



TETR_TO_HEX PROC NEAR
                and     AL, 0Fh
                cmp     AL, 09h
```

```
                jbe     NEXT
                add     AL, 07h
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, 04h
                shr     AL, CL
                call    TETR_TO_HEX
                pop     CX
                ret
BYTE_TO_HEX ENDP


WRD_TO_HEX PROC NEAR
                push    AX
                push    BX
                push    DI
                mov     BH, AH
                call    BYTE_TO_HEX
                mov     DS:[DI], AH
                dec     DI
                mov     DS:[DI], AL
                dec     DI
                mov     AL, BH
                call    BYTE_TO_HEX
                mov     DS:[DI], AH
                dec     DI
                mov     DS:[DI], AL
                pop     DI
                pop     BX
                pop     AX
                ret
WRD_TO_HEX ENDP


BYTE_TO_DEC PROC NEAR
                push    AX
                push    CX
                push    DX
                push    SI
                xor     AH, AH
                xor     DX, DX
                mov     CX, 0Ah
loop_bd:        div     CX
                or      DL, 30h
                mov     DS:[SI], DL
                dec     SI
                xor     DX, DX
                cmp     AX, 0Ah
                jae     loop_bd
                cmp     AL, 00h
                je      end_l
                or      AL, 30h
                mov     DS:[SI], AL
```

```
end_l:          pop     SI
                pop     DX
                pop     CX
                pop     AX
                ret
BYTE_TO_DEC ENDP

WRD_TO_DEC PROC NEAR
                push    BX
                xor     BX, BX
                call    DWRD_TO_DEC
                pop     BX
                ret
WRD_TO_DEC ENDP


DWRD_TO_DEC PROC NEAR
                push    AX
                push    BX
                push    CX
                push    DX
                push    DI
                jmp     clear_dd
cont_dd:        mov     AX, CX
                mov     BX, DX
clear_dd:       xor     CX, CX
                xor     DX, DX
check_dd:       cmp     BX, 00h
                ja      subst_dd
                cmp     AX, 0Ah
                jb      write_dd
subst_dd:       clc
                sub     AX, 0Ah
                sbb     BX, 00h
                clc
                add     CX, 01h
                adc     DX, 00h
                jmp     check_dd
write_dd:       add     AX, 30h
                mov     DS:[DI], AL
                dec     DI
                test    CX, CX
                jnz     cont_dd
                test    DX, DX
                jnz     cont_dd
                pop     DI
                pop     DX
                pop     CX
                pop     BX
                pop     AX
                ret
DWRD_TO_DEC ENDP

PR_CHR_BIOS PROC NEAR
                push    AX
                push    BX
                push    CX
                xchg    AL, CL
                mov     AH, 0Fh
                int     10h
                xchg    AL, CL
                mov     AH, 09h
                mov     CX, 01h
```

```
                int     10h
                pop     CX
                pop     BX
                pop     AX
                ret
PR_CHR_BIOS ENDP


PR_STR_BIOS PROC NEAR
                push    AX
                push    BX
                push    CX
                push    DX
                push    DI
                push    ES
                mov     AX, DS
                mov     ES, AX
                mov     AH, 0Fh
                int     10h
                mov     AH, 03h
                int     10h
                mov     DI, 00h
dsbp_nxt:       cmp     byte ptr DS:[BP+DI], CHR_EOT
                je      dsbp_out
                inc     DI
                jmp     dsbp_nxt
dsbp_out:       mov     CX, DI
                mov     AH, 13h
                mov     AL, 01h
                int     10h
                pop     ES
                pop     DI
                pop     DX
                pop     CX
                pop     BX
                pop     AX
                ret
PR_STR_BIOS ENDP


INT_09H_PRO PROC FAR
                push    AX
                push    BX
                push    CX
                push    BP
                push    DI
                push    DS
                push    ES
                mov     AX, L5_DATA
                mov     DS, AX
                mov     AX, 40h
                mov     ES, AX
                inc     IT_CNTR
                mov     AL, ES:[17h]
                and     AL, 00001100b
                cmp     AL, 00001100b
                jne     orig_int
                in      AL, 60h
                jmp     chk_hk01
orig_int:       pushf
                call    dword ptr DS:[KEEP_IP]
                jmp     int_quit
chk_hk01:       cmp     AL, HK_KEY1
```

```
                jne     chk_hk02
                mov     AH, AL
                in      AL, 61h
                or      AL, 10000000b
                out     61h, AL
                and     AL, 01111111b
                out     61h, AL
                mov     BL, INF_CLR
                lea     BP, INF_USG
                call    PR_STR_BIOS
                lea     BP, INF_HK1
                call    PR_STR_BIOS
                lea     BP, INF_HK2
                call    PR_STR_BIOS
                mov     CH, AH
                mov     CL, HK_ASC1
                jmp     buff_wrt
chk_hk02:       cmp     AL, HK_KEY2
                jne     orig_int
                mov     AH, AL
                in      AL, 61h
                or      AL, 10000000b
                out     61h, AL
                and     AL, 01111111b
                out     61h, AL
                lea     DI, INT_CNT
                add     DI, 57
                mov     AX, IT_CNTR
                call    WRD_TO_DEC
                mov     BL, INF_CLR
                lea     BP, INT_CNT
                call    PR_STR_BIOS
                mov     CH, AH
                mov     CL, HK_ASC2
                jmp     buff_wrt
buff_wrt:       mov     AH, 05h
                int     16h
                cmp     AL, 00h
                jne     int_quit
                jmp     int_quit
int_quit:       mov     AL, 20h
                out     20h, AL
                pop     ES
                pop     DS
                pop     DI
                pop     BP
                pop     CX
                pop     BX
                pop     AX
                iret
INT_09H_PRO ENDP


l5_start:       mov     BX, L5_DATA
                mov     DS, BX
                push    ES
                mov     AH, 35h
                mov     AL, 09h
                int     21h
                mov     KEEP_CS, ES
                mov     KEEP_IP, BX
                pop     ES
                jmp     cmds_buf
```

```
cmds_buf:       xor     BH, BH
                xor     CH, CH
                mov     CL, ES:[80h]
                cmp     CL, 00h
                je      sign_chk
                lea     DI, CMD_BUF
                mov     SI, 81h
                mov     AH, 00h
                jmp     cmds_chk
cmds_chk:       mov     AL, byte ptr ES:[SI]
                cmp     AL, '"'
                jne     cmds_chr
                not     AH
                and     AH, 00000001b
                jmp     cmds_nxt
cmds_chr:       cmp     AL, ' '
                jne     cmds_wrt
                cmp     AH, 00h
                jne     cmds_wrt
                mov     AL, 01h
                jmp     cmds_wrt
cmds_wrt:       mov     DS:[DI], AL
                inc     DI
                jmp     cmds_nxt
cmds_nxt:       inc     SI
                loop    cmds_chk
                mov     AL, 01h
                mov     DS:[DI], AL
                cmp     AH, 00h
                jne     cmds_err
                lea     DI, CMD_BUF
                jmp     pars_chk
cmds_err:       mov     BL, INF_CLR
                lea     BP, CMD_ERR
                call    PR_STR_BIOS
                jmp     dos_quit


sign_chk:       mov     AX, L5_DATA
                sub     AX, L5_CODE
                mov     CX, KEEP_CS
                add     CX, AX
                mov     ES, CX
                lea     DI, L5_SIGN
                lea     CX, CMD_ERR
                sub     CX, DI
                xor     BL, BL
                jmp     sign_nxt
sign_nxt:       mov     AL, DS:[DI]
                mov     AH, ES:[DI]
                cmp     AL, AH
                jne     cint_chk
                inc     DI
                loop    sign_nxt
                mov     BL, 01h
                jmp     cint_chk


pars_chk:       cmp     byte ptr DS:[DI], 00h
                je      sign_chk
                cmp     byte ptr DS:[DI], 01h
```

```
                je      pars_nxt
                jmp     pars_st1
pars_st1:       cmp     byte ptr DS:[DI], '/'
                je      pars_st2
                cmp     byte ptr DS:[DI], '\'
                je      pars_st2
                jmp     pars_unk
pars_st2:       inc     DI
                cmp     byte ptr DS:[DI], 'u'
                je      pars_st3
                cmp     byte ptr DS:[DI], 'U'
                je      pars_st3
                jmp     pars_unk
pars_st3:       inc     DI
                cmp     byte ptr DS:[DI], 'n'
                je      pars_ex1
                cmp     byte ptr DS:[DI], 'N'
                je      pars_ex1
                jmp     pars_unk
pars_ex1:       mov     BH, 01h
                jmp     pars_nxt
pars_nxt:       inc     DI
                jmp     pars_chk
pars_unk:       mov     BL, INF_CLR
                lea     BP, CMD_ERR
                call    PR_STR_BIOS
                jmp     dos_quit


cint_chk:       cmp     BL, 00h
                jne     cint_unf
                cmp     BH, 00h
                je      cint_inj
                mov     BL, INF_CLR
                lea     BP, UNL_ERR
                call    PR_STR_BIOS
                jmp     dos_quit
cint_inj:       push    DS
                mov     AX, seg INT_09H_PRO
                mov     DS, AX
                lea     DX, INT_09H_PRO
                mov     AH, 25h
                mov     AL, 09h
                int     21h
                pop     DS
                mov     BL, INF_CLR
                lea     BP, STA_LOA
                call    PR_STR_BIOS
                mov     BL, INF_CLR
                mov     BL, INF_CLR
            mov     BL, INF_CLR
                lea     BP, INF_USG
                call    PR_STR_BIOS
                lea     BP, INF_HK1
                call    PR_STR_BIOS
                lea     BP, INF_HK2
                call    PR_STR_BIOS
                jmp     cint_res
cint_res:       mov     AH, 01h
                int     21h
                mov     DX, L5_STACK
                add     DX, STK_SIZ
                sub     DX, L5_CODE
```

```
                add     DX, PSP_SIZ
                xor     AL, AL
                mov     AH, 31h
                int     21h
                jmp     dos_quit
cint_unf:       cmp     BH, 00h
                jne     cint_unl
                mov     BL, INF_CLR
                lea     BP, STA_ALR
                call    PR_STR_BIOS
                mov     BL, INF_CLR
                mov     BL, INF_CLR
            mov     BL, INF_CLR
                lea     BP, INF_USG
                call    PR_STR_BIOS
                lea     BP, INF_HK1
                call    PR_STR_BIOS
                lea     BP, INF_HK2
                call    PR_STR_BIOS
                jmp     dos_quit
cint_unl:       mov     BL,INF_CLR
                lea     BP, STA_ALR
                call    PR_STR_BIOS
                cli
                push    DS
                mov     AX, ES:[KEEP_CS]
                mov     DS, AX
                mov     DX, ES:[KEEP_IP]
                mov     AH, 25h
                mov     AL, 09h
                int     21h
                pop     DS
                sti
                mov     AX, KEEP_CS
                sub     AX, PSP_SIZ
                mov     ES, AX
                mov     BX, ES:[2Ch]
                mov     AH, 49h
                int     21h
                jc      cint_per
                mov     ES, BX
                mov     AH, 49h
                int     21h
                jc      cint_eer
                mov     BL, INF_CLR
                lea     BP, STA_UNL
                call    PR_STR_BIOS
                jmp     dos_quit
cint_per:       lea     DI, PRM_ERR
                add     DI, 64
                call    WRD_TO_HEX
                mov     BL,INF_CLR
                lea     BP, PRM_ERR
                call    PR_STR_BIOS
                jmp     dos_quit
cint_eer:       lea     DI, ENM_ERR
                add     DI, 64
                call    WRD_TO_HEX
                mov     BL, INF_CLR
                lea     BP, ENM_ERR
                call    PR_STR_BIOS
                jmp     dos_quit
```

```
dos_quit:       mov     AH, 01h
                int     21h
                mov     AH, 4Ch
                int     21h

L5_CODE ENDS
END START
```