

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

ОТЧЕТ
по лабораторной работе №5
по дисциплине «Организация ЭВМ и систем»
Тема: «Сопряжение стандартного и пользовательского обработчика
прерываний »

Студент гр. 8381

Преподаватель

Переверзев Д.Е.

Ефремов М.А.

Санкт-Петербург

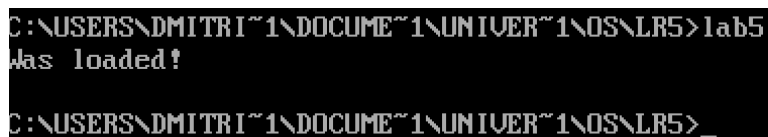
2019

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.

Выполнение работы.

Программа проверяет, установлено ли пользовательское прерывание с вектором 09h. Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено. Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход через функцию 4Ch прерывания 21h. Выгружает прерывание по соответствующему значению параметра в командной строке /un. Выгрузка прерывания состоит в восстановлении стандартного вектора прерываний и освобождения памяти, занимаемой резидентом. Осуществляется выход через функцию 4Ch прерывания 21h. Результат работы программы представлен на рис. 1.



```
C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR5>lab5
was loaded!
C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR5>_
```

Рисунок 1 – результат работы программы LAB5.EXE.

Для проверки размещения прерывания в памяти была запущена программа из лабораторной работы 3.

```

C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR5>lab3
Amount of available memory: 648128 B.
Amount of extended memory: 15360 KB.
New MCB:
Type: 4Dh. Sector: 0008h. Size:      16 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0000h. Size:      64 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0040h. Size:     256 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:     144 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:     608 B. Information in last bytes: LAB5
New MCB:
Type: 4Dh. Sector: 01C3h. Size:     144 B. Information in last bytes:
New MCB:
Type: 5Ah. Sector: 01C3h. Size: 648128 B. Information in last bytes: LAB3
C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR5>

```

Рисунок 2 – состояние памяти после загрузки собственного прерывая

После повторного запуска программы было выведено сообщение о том, что резидентная программа уже загружена. Результат повторного запуска работы представлен на рис. 3.

```

C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR5>lab5
Already loaded!
C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR5>_

```

Рисунок 3 – повторный запуск программы lab5.exe.

Была запущена программа с ключом выгрузки. Для того чтобы проверить, что память, занятая резидентом, освобождена, был выполнен запуск программы лабораторной работы No3.

```

C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR5>lab5 /un
Was unloaded!
C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR5>_

```

Рисунок 4 – Результат запуска программы с ключом выгрузки.

```
C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR5>lab3
Amount of available memory: 648912 B.
Amount of extended memory: 15360 KB.
New MCB:
Type: 4Dh. Sector: 0008h. Size:      16 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0000h. Size:      64 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0040h. Size:     256 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:     144 B. Information in last bytes:
New MCB:
Type: 5Ah. Sector: 0192h. Size: 648912 B. Information in last bytes: LAB3
C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR5>
```

Рисунок 5 – Состояние памяти после выгрузки резидентной программы.

Контрольные вопросы

1. Какого типа прерывания использовались в работе?

- аппаратные прерывания (09h)
- прерывания MS DOS (int 21h)
- прерывания BIOS (int 16h)

2. Чем отличается скан-код от кода ASCII?

- Скан код – номер, жестко закрепленный за клавишей, который посылается контроллером клавиатуры в порт 60h. Код ASCII – номер, закрепленный за символом в таблице кодировки.

Вывод

В ходе лабораторной работы был встроен пользовательский обработчик прерывания в стандартное прерывание от клавиатуры.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ. lab5.asm

_CODE SEGMENT

ASSUME CS:_CODE, DS:_DATA, ES:NOTHING, SS:_STACK

ROUT PROC FAR

jmp start

SIGNATURE dw 01984h

KEEP_PSP dw 0

KEEP_IP dw 0

KEEP_CS dw 0

INT_STACK dw 100 dup (?)

COUNT dw 0

KEEP_SS dw 0

KEEP_AX dw ?

KEEP_SP dw 0

KEY_CODE db 2ch

start:

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

in al, 60h

cmp al, KEY_CODE

je DO_REQ

```
pushf
call dword ptr CS:KEEP_IP
jmp ROUT_END
```

DO_REQ:

```
push ax
in al, 61h
mov ah, al
or al, 80h
out 61h, al
xchg ah, al
out 61h, al
mov al, 20h
out 20h, al
pop ax
```

ADD_TO_BUFF:

```
mov ah, 05h
mov cl, 02h
mov ch, 00h
int 16h
or al, al
jz ROUT_END
mov ax, 0040h
mov es, ax
mov si, 001ah
mov ax, es:[si]
mov si, 001ch
mov es:[si], ax
jmp ADD_TO_BUFF
```

ROUT_END:

```
pop di
pop dx
pop ds
pop es
pop bp
pop ax
mov AX, KEEP_SS
mov SS, AX
```

```
    mov AX, KEEP_AX
    mov SP, KEEP_SP
    mov al, 20h
    out 20h, al
    iret
```

ROUT ENDP

LAST_BYTE_ROUT:

PRINT PROC near

```
    push ax
    mov ah,09h
    int 21h
    pop ax
    ret
```

PRINT ENDP

CHECK_ROUT PROC

```
    mov ah, 35h
    mov al, 09h
    int 21h
    mov si, offset SIGNATURE
    sub si, offset ROUT
    mov ax, 01984h
    cmp ax, ES:[BX+SI]
    je ROUT_IS_LOADED
    call SET_ROUT
```

ROUT_IS_LOADED:

```
    call DELETE_ROUT
    ret
```

CHECK_ROUT ENDP

SET_ROUT PROC

```
    mov ax, KEEP_PSP
    mov es, ax
    cmp byte ptr es:[80h], 0
    je LOAD
    cmp byte ptr es:[82h], '/'
    jne LOAD
    cmp byte ptr es:[83h], 'u'
    jne LOAD
```



```
    cmp byte ptr es:[84h], 'n'
    jne LOAD
    lea dx, NOTLOADED
    call PRINT
    jmp EXIT
```

LOAD:

```
    mov ah, 35h
    mov al, 09h
    int 21h
    mov KEEP_CS, ES
    mov KEEP_IP, BX
    lea dx, LOADED
    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_ROUT
    mov cl, 4
    shr dx, cl
    inc dx
    add dx, _CODE
    sub dx, KEEP_PSP
    sub al, al
    mov ah, 31h
    int 21h
```

EXIT:

```
    sub al, al
    mov ah, 4ch
    int 21h
```

SET_ROUT ENDP

DELETE_ROUT PROC

```
    push dx
    push ax
    push ds
```

```

push es
mov ax, KEEP_PSP
mov es, ax
cmp byte ptr es:[80h], 0
je END_DELETE
cmp byte ptr es:[82h], '/'
jne END_DELETE
cmp byte ptr es:[83h], 'u'
jne END_DELETE
cmp byte ptr es:[84h], 'n'
jne END_DELETE
lea dx, UNLOADED
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 END_DELETE2
END_DELETE:
mov dx, offset ALREADYLOADED
call PRINT

```

END_DELETE2:

pop es

pop ds

pop ax

pop dx

ret

DELETE_ROUT ENDP

MAIN PROC NEAR

mov ax, _DATA

mov ds, ax

mov KEEP_PSP, es

call CHECK_ROUT

mov ax, 4C00h

int 21h

ret

MAIN ENDP

_CODE ENDS

_STACK SEGMENT STACK

db 512 dup(0)

_STACK ENDS

_DATA SEGMENT

LOADED db 'Was loaded!', 0dh, 0ah, '\$'

UNLOADED db 'Was unloaded!', 0dh, 0ah, '\$'

ALREADYLOADED db 'Already loaded!', 0dh, 0ah, '\$'

NOTLOADED db 'Was not loaded!', 0DH, 0AH, '\$'

_DATA ENDS

END MAIN