

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. 7383

Ханова Ю.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2018

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.

Ход работы.

В ходе работы были написаны функции, их описание представлено в табл.1.

Таблица 1 – описание функций.

| Название процедуры | Описание |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PRINT | Выводит на экран строку |
| ROUT | Пользовательский обработчик прерывания: при нажатии клавиши Tab выводится символ 'D' |
| CHECK_INT | Проверяет, установлено ли пользовательское прерывание (с помощью проверки сигнатуры). Если в командной строке был параметр /up и сигнатуры функций прерываний совпадали, вызывается процедура DELETE_INT для выгрузки пользовательского прерывания |
| SET_INT | Устанавливает пользовательское прерывание в поле векторов прерываний |
| DELETE_INT | Удаляет пользовательское прерывание в поле векторов прерываний |

Также использовались следующие данные, представленные в табл. 2.

Таблица 2 – описание переменных.

| Название переменной | Тип | Описание |
|----------------------------|------------|-----------------------------|
| ALREADY_LOADED | db | Строка-сообщение для вывода |

| | | |
|-----------|----|--------------------------------------------------------------------------------------------------------------------------|
| | | информации о том, что пользовательское прерывание уже установлено |
| UNLOADED | db | Строка-сообщение для вывода информации о том, что пользовательское прерывание выгружено |
| IS_LOADED | db | Строка-сообщение для вывода информации о том, что пользовательское прерывание установлено |
| SIGNATURE | db | Строка-сигнатура, которая используется для определения: было ли установлено пользовательское прерывание или нет |
| KEEP_CS | dw | Переменная для сохранения значения в регистре CS |
| KEEP_IP | dw | Переменная для сохранения значения в регистре IP |
| KEEP_PSP | dw | Переменная для сохранения сегментного адреса PSP |
| KEEP_SS | dw | Переменная для сохранения сегментного адреса стека |
| KEEP_SP | dw | Переменная для сохранения указателя стека |
| KEEP_AX | dw | Переменная для сохранения значения в регистре AX |

1. Оценим состояние памяти до запуска программы созданной программы
lr5.exe с помощью программы lr3_1.com:

```

C:\>lr3_1.com
Amount of available memory: 648912
Extended memory size:15360
Chain of MCB:
Address  Type MCB  Address PSP    Size    SC/SD
016F    004D    0008         16
0171    004D    0000         64
0176    004D    0040        256
0187    004D    0192        144
0191    005A    0192    648912    LR3_1

```

Рис. 1. Результат работы программы lr3_1.com.

2. Запуск программы lr5.exe:

```
C:\>lr5.exe
User interruption is loaded!
```

Рис. 2. Результат работы программы lr5.exe.

Как видно на Рис. 2, резидентный обработчик прерывания 09h установлен.

3. Запустим программу lr5.exe ещё раз:

```
C:\>lr5.exe
User interruption is already loaded!
```

Рис. 3. Результат повторного запуска программы lr5.exe.

Можно убедиться, в том, что программа определяет установленный обработчик прерываний.

4. Проверим загрузку пользовательского обработчика прерывания и его работы с помощью нажатия клавиши Tab и других различных символов:

```
C:\>DDDDD rgesca
```

Рис. 4. Результат ввода различных символов.

На Рис. 4 видно, что при нажатии клавиши Tab на экран выводится символ 'D'. Это значит, что установленное пользовательское прерывание работает корректно.

5. Проверим размещение прерывания в памяти с помощью программы lr3_1.com, которая отображает карту памяти в виде списка блоков MCB:

```
C:\>lr3_1.com
Amount of available memory: 648208
Extended memory size:15360
Chain of MCB:
Address  Type MCB  Address PSP   Size   SC/SD
016F     004D    0008        16
0171     004D    0000        64   DPMILOAD
0176     004D    0040       256
0187     004D    0192       144
0191     004D    0192       528   LR5
01B3     004D    01BE       144
01BD     005A    01BE     648208   LR3_1
```

Рис. 5. Результат работы программы lr3_1.com после запуска программы lr5.exe.

6. Запустим программу lr5.exe с ключом выгрузки /un:

```
C:\>lr5.exe /un
User interruption is unloaded!
```

Рис. 6. Результат работы программы lr5.exe, запущенной с ключом /un.

Очевидно, что резидентный обработчик прерывания был выгружен: об этом нас информирует сообщение, выведенное на экран программой.

7. Убедимся в том, что занятая резидентом память освобождена, запустив программу lr3_1.com:

```
C:\>lr3_1.com
Amount of available memory: 648912
Extended memory size:15360
Chain of MCB:
Address  Type MCB  Address PSP    Size    SC/SD
016F     004D      0008         16
0171     004D      0000         64    DPMILOAD
0176     004D      0040        256
0187     004D      0192        144
0191     005A      0192    648912    LR3_1
```

Рис. 7. Результат работы программы lr3_1.com выгрузки пользовательского прерывания.

Ответы на контрольные вопросы.

1. Какого типа прерывания использовались в работе?

В данной лабораторной работе использовались: аппаратные прерывания (int 09h), программные прерывания (int 21h).

2. Чем отличается скан-код от кода ASCII?

Скан-код — это код, присвоенный каждой клавише. ASCII код — это код каждого конкретного символа в соответствии со стандартной кодировочной таблицей.

Вывод.

В ходе данной лабораторной работы была исследована возможность встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры. Код программы представлен в приложении А.

ПРИЛОЖЕНИЕ А:

КОД ПРОГРАММЫ LR5.EXE

```
INT_STACK SEGMENT STACK
    DW 64 DUP (?)
INT_STACK ENDS
;-----
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, ES:DATA, SS:STACK
START: JMP MAIN
;-----
ROUT PROC FAR ;обработчик прерывания
    jmp ROUT_CODE
ROUT_DATA:
    SIGNATURE DB '0000' ;сигнатура, некоторый код, который
идентифицирует резидент
    KEEP_IP DW 0 ;и смещения прерывания
    KEEP_CS DW 0 ;для хранения сегмента
    KEEP_PSP DW 0 ;и PSP
    KEEP_SS DW 0
    KEEP_AX DW 0
    KEEP_SP DW 0
ROUT_CODE:
    mov KEEP_AX, AX ;сохраняем ax
    mov KEEP_SS, SS ;сохраняем стек
    mov KEEP_SP, SP
    mov AX, seg INT_STACK ;устанавливаем собственный стек
    mov SS, AX
    mov SP, 64h
    mov AX, KEEP_AX
    push AX ;сохранение изменяемых регистров
    push DX
    push DS
    push ES

    ;проверяем скан-код клавиши
    in AL, 60h ;читать ключ
    cmp AL, 0Fh ;это требуемый код? 0F - скан-код Tab
    je DO_REQ ;получили требуемый скан-код

    ;стандартный обработчик прерывания
    pushf
    call dword ptr CS:KEEP_IP ;переход на первоначальный обработчик
    jmp ROUT_END

DO_REQ: ;отработка аппаратного прерывания
    push AX
    in AL, 61h ;взять значение порта управления клавиатурой
    mov AH, AL ;сохранить его
    or AL, 80h ;установить бит разрешения для клавиатуры
```

```

out 61h, AL ;и вывести его в управляющий порт
xchg AH, AL ;извлечь исходное значение порта
out 61h, AL ;и записать его обратно
mov AL, 20h ;послать сигнал "конец прерывания"
out 20h, AL ;контроллеру прерываний 8259
pop AX

```

```

ADD_TO_BUFF: ;запись символа в буфер клавиатуры
mov AH, 05h ;код функции
mov CL, 'D' ;пишем символ в буфер клавиатуры
mov CH, 00h
int 16h
or AL, AL ;проверка переполнения буфера
jz ROUT_END ;переполнение
;очищаем буфер клавиатуры
CLI ;запрещение прерывания
mov ax,es:[1Ah] ;берём адрес начала буфера
mov es:[1Ch],ax ;помещаем адрес начала буфера в адрес конца
STI ;разрешение прерывания
jmp ADD_TO_BUFF ;повторить

```

```

ROUT_END:
pop ES ;восстановление регистров
pop DS
pop DX
pop AX
mov SS, KEEP_SS
mov SP, KEEP_SP
mov AX, KEEP_AX
mov AL,20h
out 20h,AL
iret

```

LAST_BYTE:

ROUT ENDP

;-----

CHECK_INT PROC ;проверка прерывания

;проверка, установлено ли пользовательское прерывание с вектором 09h

```

mov AH,35h ;функция 35h прерывания 21h
mov AL,09h ;AL = номер прерывания
int 21h ;дать вектор прерывания
;выход: ES:BX = адрес обработчика прерывания

```

mov SI, offset SIGNATURE ;на определённом, известном смещении в теле резидента располагается сигнатура,

;некоторый код, который

идентифицирует резидент

sub SI, offset ROUT ;SI = смещение SIGNATURE относительно начала функции прерывания

```

mov AX,'00' ;сравнив известное значение сигнатуры

```



```

    cmp AX,ES:[BX+SI] ;с реальным кодом, находящимся в резиденте
    jne NOT_LOADED ;если значения не совпадают, то резидент не
установлен
    cmp AX,ES:[BX+SI+2] ;с реальным кодом, находящимся в резиденте
    jne NOT_LOADED ;если значения не совпадают, то резидент не
установлен
    jmp LOADED ;если значения совпадают, то резидент установлен

NOT_LOADED: ;установка пользовательского прерывания
    call SET_INT ;установка пользовательского прерывания
    ;вычисление необходимого количества памяти для резидентной
программы
    mov DX,offset LAST_BYTE ;кладём в dx размер части сегмента CODE,
содержащей пользовательское прерывание
    ;и необходимые код и данные для
него
    mov CL,4 ;перевод в параграфы
    shr DX,CL
    inc DX ;размер в параграфах
    add DX,CODE ;прибавляем адрес сегмента CODE
    sub DX,CS:KEEP_PSP ;вычитаем адрес сегмента PSP
    xor AL,AL
    mov AH,31h ;номер функции 31h прерывания 21h
    int 21h ;оставляем нужное количество памяти
    ;(dx - количество параграфов) и выходим в DOS, оставляя
программу в памяти резидентно

LOADED: ;смотрим, есть ли в хвосте /un , тогда нужно выгрузить
    push ES
    push AX
    mov AX,KEEP_PSP
    mov ES,AX
    cmp byte ptr ES:[82h], '/' ;сравниваем аргументы
    jne NOT_UNLOAD ;не совпадают
    cmp byte ptr ES:[83h], 'u' ;сравниваем аргументы
    jne NOT_UNLOAD ;не совпадают
    cmp byte ptr ES:[84h], 'n' ;сравниваем аргументы
    je UNLOAD ;совпадают

NOT_UNLOAD: ;если не /un
    pop AX
    pop ES
    mov dx,offset ALREADY_LOADED
    call PRINT
    ret
    ;выгрузка пользовательского прерывания
UNLOAD: ;если /un
    pop AX
    pop ES
    call DELETE_INT ;выгрузка пользовательского прерывания
    mov dx,offset UNLOADED ;вывод сообщения

```

```

        call PRINT
        ret
CHECK_INT ENDP
;-----
SET_INT PROC ;установка написанного прерывания в поле векторов прерываний
    push DS
    mov AH,35h ;функция получения вектора
    mov AL,09h ;номер вектора
    int 21h
    mov CS:KEEP_IP,BX ;запоминание смещения
    mov CS:KEEP_CS,ES ;и сегмента
    mov DX,offset ROUT ;смещение для процедуры в DX
    mov AX,seg ROUT ;сегмент процедуры
    mov DS,AX ;помещаем в DS
    mov AH,25h ;функция установки вектора
    mov AL,09h ;номер вестора
    int 21h ;меняем прерывание
    pop DS
    push DX
    mov DX,offset IS_LOADED ;вывод сообщения
    call PRINT
    pop DX
    ret
SET_INT ENDP
;-----
DELETE_INT PROC ;удаление написанного прерывания в поле векторов
прерываний
    push DS
    ;восстановление вектора прерывания
    CLI ;запрещение прерывания
    mov DX,ES:[BX+SI+4] ;KEEP_IP
    mov AX,ES:[BX+SI+6] ;KEEP_CS
    mov DS,AX ;DS:DX = вектор прерывания: адрес программы обработки
прерывания
    mov AH,25h ;функция 25h прерывания 21h
    mov AL,09h ;AL = номер вектора прерывания
    int 21h ;восстанавливаем вектор
    ;освобождение памяти, занимаемой резидентом
    push ES
    mov AX, ES:[BX+SI+8] ;KEEP_PSP
    mov ES, AX
    mov ES, ES:[2Ch] ;ES = сегментный адрес (параграф) освобождаемого
блока памяти
    mov AH, 49h ;функция 49h прерывания 21h
    int 21h ;освобождение распределенного блока памяти
    pop ES
    mov ES, ES:[BX+SI+8];KEEP_PSP ES = сегментный адрес (параграф)
освобождаемого блока памяти
    mov AH, 49h ;функция 49h прерывания 21h
    int 21h ;освобождение распределенного блока памяти
    STI ;разрешение прерывания

```

```

        pop DS
        ret
DELETE_INT ENDP
;-----
PRINT PROC NEAR ;печать на экран
        push ax
        mov ah, 09h
        int 21h
        pop ax
        ret
PRINT ENDP
;-----
MAIN:
        mov AX,DATA
        mov DS,AX
        mov CS:KEEP_PSP,ES ;сохранение PSP
        call CHECK_INT ;проверка прерывания
        xor AL,AL
        mov AH,4Ch ;выход
        int 21H
CODE ENDS
;-----
STACK SEGMENT STACK
        DW 64 DUP (?)
STACK ENDS
;-----
DATA SEGMENT
        ALREADY_LOADED DB 'User interruption is already
loaded! ',0DH,0AH,'$'
        UNLOADED DB 'User interruption is unloaded! ',0DH,0AH,'$'
        IS_LOADED DB 'User interruption is loaded! ',0DH,0AH,'$'
DATA ENDS
;-----
END START

```