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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Операционные системы»

Тема: Сопряжение стандартного и пользовательского обработчиков прерываний

Студентка гр. 7383 Маркова А.В.

Преподаватель Ефремов М.А.

Санкт-Петербург

2019

Постановка задачи.

Изучить возможность 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 определённые действия, если скан-код совпадает с определёнными кодами, которые он должен обрабатывать. Если скан- код не совпадает с этими кодами, то управление передаётся стандартному прерыванию.

Описание процедур, используемых в работе:

- LINE_OUTPUT – выводит сообщение на экран;
- ROUT – резидентный обработчик прерываний от клавиатуры. При нажатии клавиши Alt Left выводит символ «□», ASCII- код которого равен 127;
- CHECK_INT – проверка, установлено ли пользовательское прерывание с вектором 09h. Также смотрит есть ли в хвосте «/un», если да, то вызывает процедуру удаления резидента;
- DELETE – при вызове программы с ключом «/un» восстанавливает стандартный вектор прерывания и выгружает из памяти пользовательское прерывание;
- MY_INT – установка написанного прерывания в поле векторов прерываний;
- BEGINNING – главная процедура.

Таблица 1 – Описание структуры данных управляющей программы:

Название	**Тип**	**Назначение**
interrupt_already_loaded	db	Строка – сообщение для вывода информации о том, что пользовательское прерывание уже было установлено
interrupt_was_unloaded	db	Строка – сообщение для вывода информации о том, что пользовательское прерывание выгружено
interrupt_was_loaded	db	Строка – сообщение для вывода информации о том, что пользовательское прерывание установлено

Таблица 2 – Описание структуры данных собственного прерывания:

Название	**Тип**	**Назначение**
signature	db	Сигнатура пользовательского прерывания
keep_psp	dw	Переменная для сохранения сегментного адреса PSP
keep_ss	dw	Переменная для сохранения сегментного адреса стека
keep_ax	dw	Переменная для сохранения в AX
keep_sp	dw	Переменная для сохранения указателя стека
keep_cs	dw	Переменная для сохранения в CS
keep_ip	dw	Переменная для сохранения в IP
REQ_KEY	db	Скан- код Alt Left
MY_STACK	dw	Собственный стек
END_STACK	dw	Конец стека

Ход работы.

1. Был написан программный модуль .EXE, который выполняет следующие функции:
   - Проверяет установлено ли пользовательское прерывание с вектором 09h;
   - Устанавливает резидентную функцию для обработки прерывания, настраивает вектор прерывания, если прерывание не установлено;
   - Если прерывание установлено, то выводит соответствующее сообщение на экран;
   - Выгрузка прерывания происходит при получении параметра «/un» в командной строке при вызове программы.
2. Смонтирован виртуальный диск k с каталогом tasm.
3. Было произведено транслирование программы с помощью строки tasm «имя файла», в последствии чего создался объектный файл.
4. Линковка загрузочного модуля с помощью команды tlink. Результат представлен на рис. 1.

```
Z:\>mount k d:/7383/lr5/tasm
Drive K is mounted as local directory d:/7383/lr5/tasm\

Z:\>k:

K:\>tasm lr5
Turbo Assembler  Version 3.1  Copyright (c) 1988, 1992 Borland International

Assembling file:   lr5.ASM
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  471k


K:\>tlink lr5
Turbo Link  Version 5.1 Copyright (c) 1992 Borland International
```

Рисунок 1 – Результат сборки программного модуля

5. Получен LR5.EXE модуль из исходного кода для .EXE модуля.
6. Была дана оценка состояния памяти до запуска LR5.EXE, используя программу из предыдущей лабораторной работы. Вывод LR3_1.COM показан на рис. 2.

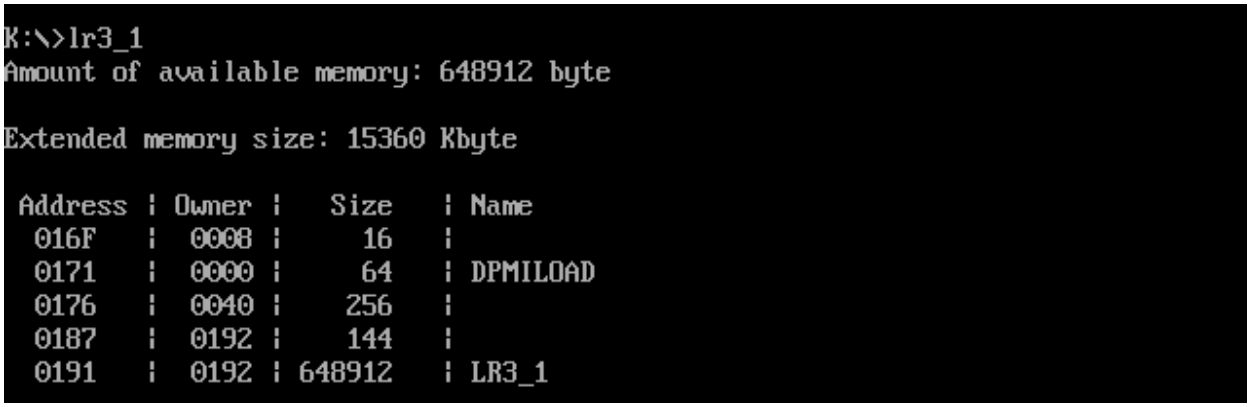

Рисунок 2 – Итог работы программы lr3_1.com

7. Запущена отлаженная программа, чтобы удостовериться, что резидентный обработчик прерывания 09h установлен. Результат запуска представлен на рис. 3.

Рисунок 3 – Результат запуска программы lr5.exe

8. Была запущена отлаженная программа ещё один раз, чтобы убедиться, что LR5.EXE определяет установленный обработчик прерывания. Выведенная информация показана на рис. 4.

Рисунок 4 – Повторный запуск программы

9. Проверено размещение прерывания в памяти с помощью LR3_1.COM, которая отображает карту памяти в виде списка блоков MCB. Вывод на консоль после запуска программы представлен на рис. 5.

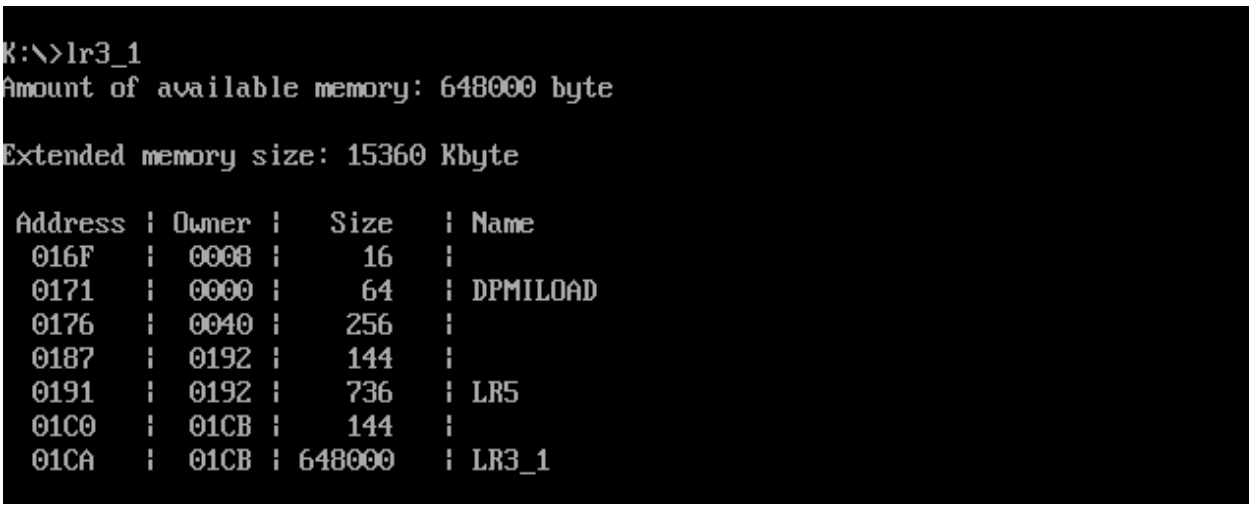

Рисунок 5 – Размещение прерывания в памяти

10. Проверим работу пользовательского обработчика прерывания с помощью нажатия клавиши Alt Left и других символов. Результат показан на рис. 6.

Рисунок 6 – Проверка изменения поведения нажатия клавиши

Видим, что при нажатии клавиши Alt Left на клавиатуре выводится заданный символ.

11. Запущена отлаженная программа с ключом выгрузки «/un», по результатам, которые представлены на рис. 7 – 8, видно, что резидентный обработчик прерывания выгружен, то есть при нажатии клавиши Alt Left ничего не выводится, а память занятая резидентом освобождена.

Рисунок 7 – Результат запуска программы с ключом «/un»

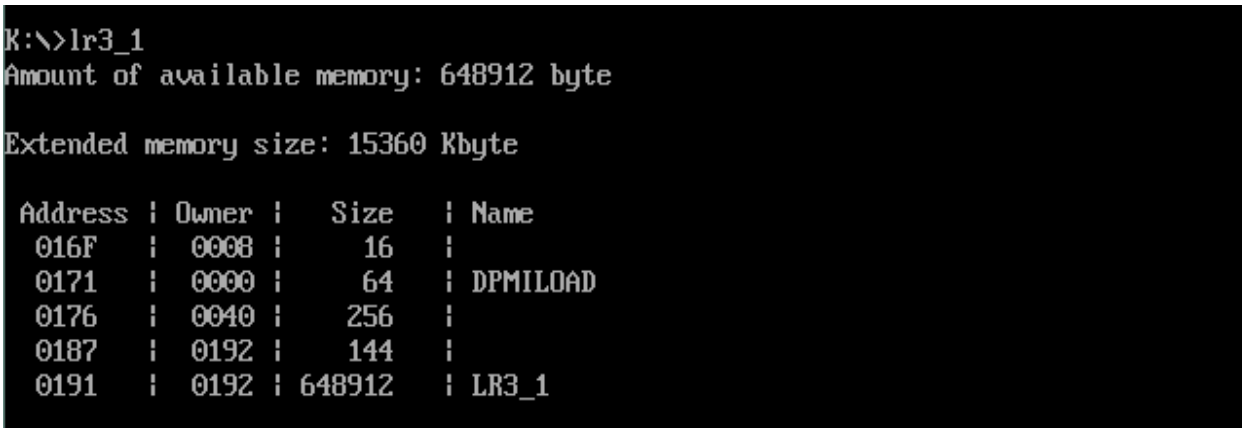

Рисунок 8 – Результат работы программы lr3_1.com

Выводы.

В ходе данной лабораторной работы была изучена возможность встраивания пользовательского обработчика прерываний в стандартный. Также был создан программный модуль, устанавливающий пользовательский обработчик прерываний от клавиатуры, который проверяет, установлено ли пользовательское прерывание с вектором 09h, устанавливает резидентную функцию для обработки прерывания, выгружает пользовательское прерывание по соответствующему значению параметра командной строки «/un», а также при нажатии определённой клавиши выводит заданный символ. Код программы представлен в приложении А.

Ответы на контрольные вопросы.

1. Какого типа прерывания использовались в работе?

В данной лабораторной работе использовались аппаратные и программные прерывания.

Аппаратное прерывание:

- int 09h – пользовательское прерывание выполняется при каждом нажатии и отпускании клавиши.

Программные прерывания:

- int 21h – используется для большинства функций DOS;
- int 16h – интерфейс прикладного уровня с клавиатурой. Нажатия клавиш на самом деле обрабатываются асинхронно на заднем плане. Когда клавиша получена от клавиатуры, она обрабатывается прерыванием int 09h и помещается в циклическую очередь.

2. Чем отличается скан- код от кода ASCII?

Скан-код – код, присвоенный каждой клавиши, с помощью которого драйвер клавиатуры распознаёт, какая клавиша была нажата. ASCII-код – уникальный числовой код в соответствии со стандартной кодировочной таблицей, присвоенный некоторым распространённым печатным и непечатным символам. Таким образом, ASCII символы не связаны напрямую с клавиатурой, обработчик прерываний от клавиатуры может по-разному обрабатывать скан-коды, в том числе и записывать ASCII-код в буфер клавиатуры.

ПРИЛОЖЕНИЕ А

LR5.ASM

```
.286

EOFLine EQU '$'                                 ; определение символьной константы
                                                ; $ - "конец строки"

ASTACK SEGMENT STACK
    DW 64 DUP (?)
ASTACK ENDS
;ДАННЫЕ
;-----------------------------------------------------------------------------------------
DATA SEGMENT
    interrupt_already_loaded DB 'Interrupt already loaded!',  0DH,0AH,EOFLine
    interrupt_was_unloaded   DB 'Interrupt was unloaded!',    0DH,0AH,EOFLine
    interrupt_was_loaded     DB 'Interrupt was loaded!',      0DH,0AH,EOFLine
DATA ENDS
;-----------------------------------------------------------------------------------------
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, ES:DATA, SS:ASTACK
START: JMP BEGINNING                            ;переход на метку
;ПРОЦЕДУРЫ
;-----------------------------------------------------------------------------------------
LINE_OUTPUT PROC NEAR                           ;вывод строки
    mov AH, 0009h
    int 21h
    ret
LINE_OUTPUT ENDP

ROUT PROC FAR                                   ;обработчик прерывания
    jmp ROUT_BEGINNING
;ДАННЫЕ
;-----------------------------------------------------------------------------------------
    signature db '0000'                         ;идентификация резидента
    keep_ip   dw 0                              ;для хранения смещения прерывания
    keep_cs   dw 0                              ;для хранения сегмента кода
    keep_psp  dw 0                              ;для хранения PSP
    keep_ss   dw 0                              ;для хранения сегмента стека
    keep_ax   dw 0                              ;для хранения регистра AX
    keep_sp   dw 0                              ;для хранения регистра SP
    REQ_KEY   db 38h                            ;скан-код Alt Left
    MY_STACK  dw 64 DUP(?)
    END_STACK dw 0
;-----------------------------------------------------------------------------------------
    ROUT_BEGINNING:
        mov  keep_ax, AX                        ;запоминаем ax
        mov  keep_ss, SS
        mov  keep_sp, SP
        mov  AX,  CS                            ;установка своего стека
        mov  SS,  AX
        mov  SP,  offset END_STACK
        mov  AX,  keep_ax
        pusha                                   ;поместить в стек значения всех 16-битных регистров общего назначения
        in   AL,  60h                           ;читать скан-код клавиши (её порядковый номер), ввод значения из порта ввода-вывода
        cmp  AL,  REQ_KEY                       ;это требуемый код?
        je   DO_PROCESSING                      ;да, активизировать обработку REQ_KEY | нет, уйти на исходный обработчик
        call dword ptr CS: keep_ip              ;переход на первоначальный обработчик
        jmp  EXIT
```

```
    DO_PROCESSING:                          ;следующий код необходим для отработки аппаратного прерывания
        push AX
        in   AL,  61h                       ;взять значение порта управления клавиатуры
        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, позволяет обменять содержимое двух операндов
        out  61h, AL                        ;и записать его обратно
        mov  AL,  20h                       ;послать сигнал "конец прерывания"
        out  20h, AL                        ;контроллеру прерываний 8259
        pop  AX

    SKIP:                                   ;записать символ в буфер клавиатуры
        mov  CL,  127                       ;аски-код □
        mov  AH,  05h                       ;код функции
        and  CH,  00h
        int  16h
        or   AL,  AL                        ;проверка на переволнение буфера
        jz   EXIT                           ;если переполнение, то очищаем буфер клавиатуры
        cli
        mov  AX,  ES:[1Ah]                  ;взятие адреса начала буфера
        mov  ES:  [1Ch], AX                 ;записываем адрес начала в конец
        sti                                 ;разрешение прерывания, путём изменения флага IF
        jmp  SKIP

    EXIT:
        popa                                ;восстановление всех регистров общего назначения
        mov  SS,  keep_ss
        mov  SP,  keep_sp
        mov  AX,  keep_ax
        mov  AL,  20h
        out  20h, AL
        mov  AX,  keep_ax
        iret

    LAST_BYTE:
        ROUT ENDP
;-----------------------------------------------------------------------------------------
CHECK_INT PROC                              ;проверка, установлено ли пользовательское прерывание с вектором 09h
    mov AH, 35h                             ;функция 35h прерывания 21h, даёт вектор прерывания
    mov AL, 09h
    int 21h                                 ;Выход: ES:BX = адрес обработчика прерывания
    mov SI, offset signature
    sub SI, offset ROUT                     ;SI = смещение signature относительно начала функции прерывания
    mov AX, '00'
    cmp AX, ES:[BX+SI]
    jne NOT_LOADED
    cmp AX, ES:[BX+SI+0002h]
    jne NOT_LOADED
    jmp LOADED

    NOT_LOADED:
        call MY_INT                         ;установка пользовательского прерывания
        mov  DX, offset LAST_BYTE           ;размер в байтах от начала
        mov  CL, 4                          ;перевод в параграфы
        shr  DX, CL                         ;сдвиг на 4 разряда вправо
        inc  DX
```

```
        add  DX, CODE
        sub  DX, keep_psp
        xor  AL, AL
        mov  AH, 31h                    ;номер функции 31h прерывания 21h, оставляем нужное количество памяти
        int  21h

    LOADED:                             ;смотрим, есть ли в хвосте /un , тогда нужно выгружать
        push ES
        push AX
        mov AX, CS: keep_psp
        mov ES, AX
        cmp byte ptr ES:[0082h], '/'
        jne NOT_UNLOAD
        cmp byte ptr ES:[0083h], 'u'
        jne NOT_UNLOAD
        cmp byte ptr ES:[0084h], 'n'
        je UNLOAD

    NOT_UNLOAD:
        pop  AX
        pop  ES
        mov  dx, offset interrupt_already_loaded
        call LINE_OUTPUT
        ret

    UNLOAD:
        pop  AX
        pop  ES
        call DELETE
        mov  dx, offset interrupt_was_unloaded
        call LINE_OUTPUT
        ret
CHECK_INT ENDP

DELETE PROC                             ;восстановление вектора прерывания
    push AX
    push DS
    push ES
    CLI                                 ;запрещение прерывания, путём сбрасывания флага IF
    mov  DX, ES: [BX+SI+0004h]
    mov  AX, ES: [BX+SI+0006h]
    mov  DS, AX                         ;DS:DX = вектор прерывания: адрес программы обработки прерывания
    mov  AH, 25h                        ;функция 25h прерывания 21h, устанавливает вектор прерывания
    mov  AL, 09h
    int  21h
    mov  AX, ES: [BX+SI+0008h]
    mov  ES, AX
    mov  ES, ES:[2Ch]                   ;ES = сегментный адрес освобождаемого блока памяти
    mov  AH, 49h                        ;функция 49h прерывания 21h, освободить распределённый блок памяти
    int  21h
    pop  ES
    mov  ES, ES: [BX+SI+0008h]
    mov  AH, 49h
    int  21h
    STI                                 ;разрешение прерывания
    pop  DS
    pop  AX
```

```
    ret
DELETE ENDP

;-------------------------------------------------------------------------------------------
MY_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  keep_ip, BX                            ;запоминание смещения
    mov  keep_cs, ES
    mov  DX, offset ROUT                        ;смещение для процедуры в DX
    mov  AX, seg ROUT
    mov  DS, AX
    mov  AH, 25h                                ;функция установки вектора
    mov  AL, 09h                                ;номер вектора
    int  21h
    pop  DS
    push DX
    mov  DX, offset interrupt_was_loaded
    call LINE_OUTPUT
    pop  DX
    ret
MY_INT ENDP
;-------------------------------------------------------------------------------------------
BEGINNING:
    mov  AX, DATA
    mov  DS, AX
    mov  keep_psp, ES
    call CHECK_INT
    xor  AL, AL
    mov  AH, 4Ch                                ;выход
    int  21H
CODE ENDS
    END  START
```