

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

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

Студентка гр. 7382

Чемова К.А.

Преподаватель

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.

Необходимые сведения для составления программы.

Клавиатура содержит микропроцессор, который воспринимает каждое нажатие на клавишу и посылает скан-код в порт микросхемы интерфейса с периферией. Когда скан-код поступает в порт, то вызывается аппаратное прерывание клавиатуры (int 09h). Процедура обработки этого прерывания считывает номер клавиши из порта 60h, преобразует номер клавиши в соответствующий код, выполняет установку флагов в байтах состояния, загружает номер клавиши и полученный код в буфер клавиатуры.

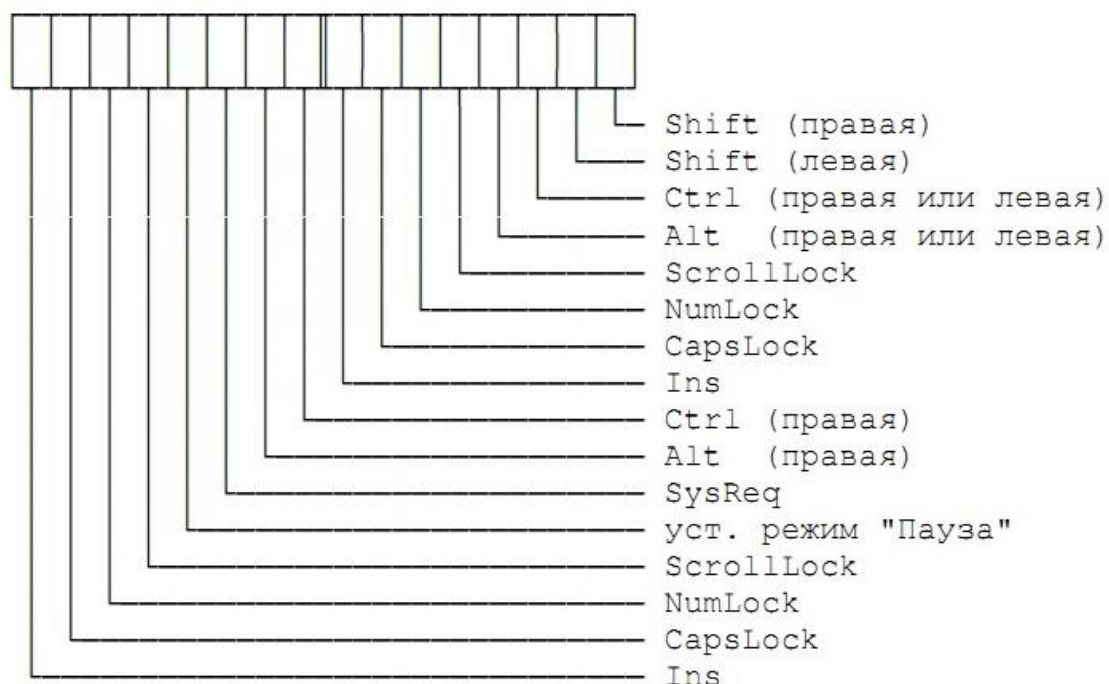
В прерывании клавиатуры можно выделить три основных шага:

- 1) Прочитать скан-код и послать клавиатуре подтверждающий сигнал.
- 2) Преобразовать скан-код в номер кода или в установку регистра статуса клавиш-переключателей.
- 3) Поместить код клавиши в буфер клавиатуры.

Текущее содержимое буфера клавиатуры определяется указателями на начало и

Адрес в памяти	Размер в байтах	Содержимое
0040:001A	2	Адрес начала буфера
0040:001C	2	Адрес конца буфера
0040:001E	32	Буфер клавиатуры
0040:0017	2	Байты состояния

Флаги в байтах состояния устанавливаются в 1, если нажата соответствующая клавиша или установлен режим. Соответствие флагов и клавиш показано ниже.



В момент вызова прерывания скан-код будет находиться в орте 60h. Поэтому сначала надо этот код прочитать командой IN и сохранить на стеке. Затем используется порт 61H, чтобы быстро послать сигнал подтверждения микропроцессору клавиатуры. Надо просто установить бит 7 в 1, а затем сразу изменить его назад в 0. Заметим, что бит 6 порта 61H управляет сигналом часов клавиатуры. Он всегда должен быть установлен в 1, иначе клавиатура будет выключена. Эти адреса портов применимы и к АТ, хотя он и не имеет микросхемы интерфейса с периферией 8255.

Сначала скан-код анализируется на предмет того, была ли клавиша нажата (код нажатия) или отпущена (код освобождения). Код освобождения состоит из двух байтов: сначала 0F0H, а затем скан-код. Все коды освобождения отбрасываются, кроме случая клавиш-переключателей, для которых делаются соответствующие изменения в байтах их статуса. С другой стороны, все коды нажатия обрабатываются. При этом опять могут изменяться байты статуса клавиш-переключателей. В случае же символьных

кодов, надо проверять байты статуса, чтобы определить, на пример, что скан-код 30 соответствует нижнему или верхнему регистру буквы А. После того как введенный символ идентифицирован, процедура ввода с клавиатуры должна найти соответствующий ему код ASCII или расширенный код. Приведенный пример слишком короток, чтобы рассмотреть все случаи. В общем случае скан-коды сопоставляются элементам таблицы данных, которая анализируется инструкцией XLAT. XLAT принимает в AL число от 0 до 255, а возвращает в AL 1-байтное значение из 256-байтной таблицы, на которую указывает DS:BX. Таблица может находиться в сегменте данных. Если в AL находился скан-код 30, то туда будет помещен из таблицы байт номер 30 (31-й байт, так как отсчет начинается с нуля). Этот байт в таблице должен быть установлен равным 97, давая код ASCII для "a". Конечно, для получения заглавной А нужна другая таблица, к которой обращение будет происходить, если статус сдвига установлен. Или заглавные буквы могут храниться в другой части той же таблицы, но в этом случае к скан-коду надо будет добавлять смещение, определяемое статусом клавиш-переключателей.

Номера кодов должны быть помещены в буфер клавиатуры. Процедура должна сначала проверить, имеется ли в буфере место для следующего символа. Буфер устроен как циклическая очередь. Ячейка памяти 0040:001A содержит указатель на голову буфера, а 0040:001C - указатель на хвост. Эти словные указатели дают смещение в области данных BIOS (которая начинается в сегменте 40H) и находятся в диапазоне от 30 до 60. Новые символы вставляются в ячейки буфера с более старшими адресами, а когда достигнута верхняя граница, то следующий символ переносится в нижний конец буфера. Когда буфер полон, то указатель хвоста на 2 меньше указателя на голову – кроме случая, когда указатель на голову равен 30 (начало области буфера), а в этом случае буфер полон, когда указатель хвоста равен 60. Для вставки символа в буфер, надо поместить его в позицию, на которую указывает хвост буфера и затем увеличить указатель хвоста на 2; если указатель хвоста был равен 60, то надо изменить его значение на 30.

Код для обработки прерывания 09H

```
push ax
in  al,60H          ;читать ключ
cmp al,REQ KEY      ;это требуемый код?
je do-req           ; да, активизировать обработку REQ KEY
; нет, уйти на исходный обработчик
pop ax
jmp cs:[int9_vect] ;переход на первоначальный обработчик do_req:
;следующий код необходим для обработки аппаратного прерывания
in al,61H ;взять значение порта управления клавиатурой

mov,ah,al or        ;сохранить его
al, 8 0h out        ;установить бит разрешения для клавиатуры
61H,al xchg         ;и вывести его в управляющий порт
ah,al              ;извлечь исходное значение порта

mov al,20H ;послать сигнал "конец прерывания"
out 20H,al      ; контроллеру прерываний 8259
; дальше - прочие проверки
```

Записать символ в буфер клавиатуры можно с помощью функции 05h прерывания 16h:

```
mov ah,05h ; Код функции
mov cl,'D' ; Пишем символ в буфер клавиатуры
mov ch,00h ;
int 16h ;
or al,al ; проверка переполнения буфера
jnz skip ; если переполнен идем skip ; работать дальше
skip: ; очистить буфер и повторить
```

Постановка задачи.

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .EXE, который выполняет такие же функции, как в программе ЛР 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 int 21h.

Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h.

Выгрузка прерывания по соответствующему значению параметра в командной строке /un. Выгрузка прерывания состоит в восстановлении стандартного вектора прерываний и освобождении памяти, занимаемой резидентом. Затем осуществляется выход по функции 4Ch прерывания int 21h.

Для того чтобы проверить установку прерывания, можно поступить следующим образом. Прочитать адрес, записанный в векторе прерывания. Предположим, что этот адрес указывает на точку входа в установленный резидент. На определенном, известном смещении в теле резидента располагается сигнатура, некоторый код, который идентифицирует резидент. Сравнив известное значение сигнатуры с реальным кодом, находящимся в резиденте, можно определить, установлен ли резидент. Если значения совпадают, то резидент установлен. Длину кода сигнатуры должна быть достаточной, чтобы сделать случайное совпадение маловероятным.

Программа должна содержать код устанавливаемого прерывания в виде удаленной процедуры. Этот код будет работать после установки при возникновении прерывания. Он должен выполнять следующие функции:

Сохранить значения регистров в стеке при входе и восстановить их при выходе.

При выполнении тела процедуры анализируется скан-код.

Если этот код совпадает с одним из заданных, то требуемый код записывается в буфер клавиатуры.

Если этот код не совпадает ни с одним из заданных, то осуществляется передача управления стандартному обработчику прерывания.

Шаг 2. Запустите отлаженную программу и убедитесь, что резидентный обработчик прерывания 09h установлен. Работа прерывания проверяется введением различных символов, обрабатываемых установленным обработчиком и стандартным обработчиком.

Шаг 3. Также необходимо проверить размещение прерывания в памяти. Для этого запустите программу ЛР 3, которая отображает карту памяти в виде с писка блоков МСВ. Полученные результаты поместите в отчет.

Шаг 4. Запустите отлаженную программу еще раз и убедитесь, что программа определяет установленный обработчик прерываний. Полученные результаты поместите в отчет.

Шаг 5. Запустите отлаженную программу с ключом выгрузки и убедитесь, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память, занятая резидентом освобождена. Для этого также следует запустить программу ЛР 3. Полученные результаты поместите в отчет.

Оформить отчёт и ответить на контрольные вопросы.

Процедуры используемые в программе.

- PRINT – выводит сообщение на экран;
- ROUT – резидентный обработчик прерываний от клавиатуры. При нажатии клавиши Alt Left выводит символ «!», ASCII-код которого равен 33;
- 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;

Структуры данных.

Таблица 2 – Структуры данных используемые в программе

Название поля данных	Тип	Назначение
int_not_loaded	db	Резидент не загружен
int_al_loaded	db	Резидент уже загружен
int_loaded	db	Резидент загружен
int_unload	db	Резидент был выгружен
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.

Проверка состояния памяти до выполнения разработанного модуля представлена на рис.1.


```

C:\>LAB3_1.COM
Available memory, B: 648912
Extended memory, KB: 15420
Address   Type MCB      PSP      Size, B      SD/SC
016F      4D      0008        16
0171      4D      0000        64
0176      4D      0040       256
0187      4D      0192       144
0191      5A      0192      648912      LAB3_1

```

Рисунок 1 – Состояние памяти до выполнения программы

Установим резидентный обработчик прерываний. Результат представлен на рис.2.

```

C:\>LAB5.EXE
Interrupt was loaded!

```

Рисунок 2 – Установка резидентного обработчика прерываний

Попробуем запустить обработчик повторно. Результат представлен на рис.3.

```

C:\>LAB5.EXE
Interrupt already loaded!

```

Рисунок 3 – Проверка установки обработчика

Проверим состояние памяти на наличие загруженного модуля. Результат представлен на рис.4.


```

C:\>LAB3_1.COM
Available memory, B: 648000
Extended memory, KB: 15420
Address   Type MCB      PSP      Size, B      SD/SC
016F      4D      0008        16
0171      4D      0000        64
0176      4D      0040       256
0187      4D      0192       144
0191      4D      0192       736      LAB5
01C0      4D      01CB       144
01CA      5A      01CB      648000      LAB3_1

```

Рисунок 4 – Проверка состояния памяти


Проверим работу пользовательского обработчика прерывания с помощью нажатия клавиши Alt Left и других символов. Результат показан на рис.5.



```
C:\>!!!qwerty!!!!123546789!!!!1245
```

Рисунок 5 – Проверка работы программы

Запустим отложенную программу с ключом /un и проверяем состояние памяти после выгрузки резидента. Результат представлен на рис.6.



```
C:\>LAB5.EXE /un  
Interrupt was unloaded!
```

Рисунок 6 – Запуск отложенной программы

Выводы.

В ходе лабораторной работы был построен пользовательский обработчик прерывания, встроенный в стандартный обработчик от клавиатуры. Изучены дополнительные функции работы с памятью, такие как: установка программы-резидента и выгрузка его из памяти, а также организация и управление прерываниями.

Ответы на контрольные вопросы.

1) Какого типа прерывания использовались в работе?

Прерывания функций DOS (21h), прерывания функций BIOS (16h, 09h).

2) Чем отличается скан код от кода ASCII?

Код ASCII – это код символа из таблицы ASCII, а скан-код – код, присвоенный каждой клавише, с помощью которого драйвер клавиатуры распознает, какая клавиша была нажата.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
.286
ASTACK SEGMENT STACK
    DW 64 DUP (?)
ASTACK ENDS
;ДАННЫЕ
;-----
DATA SEGMENT
interrupt_already_loaded db 'Interrupt already loaded!', 0DH, 0AH, '$'
interrupt_was_unloaded  db 'Interrupt was unloaded!', 0DH, 0AH, '$'
interrupt_was_loaded     db 'Interrupt was loaded!', 0DH, 0AH, '$'
DATA ENDS
;-----
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, ES:DATA, SS:ASTACK
;ПРОЦЕДУРЫ
;-----
PRINT PROC NEAR
;вывод строки
    mov AH, 0009h
    int 21h
    ret
PRINT 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     PROCESSING              ;да, активизировать обработку REQ_KEY
| нет, уйти на исходный обработчик
    call   dword ptr CS: keep_ip   ;переход на первоначальный обработчик
    jmp    EXIT
;*****
*****
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,  33      ;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:
;восстановление регистров

```

```

        popa
        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                ;даёт вектор прерывания
        mov     AL, 09h
        int     21h                    ;Выход: ES:BX = адрес обработчика
        прерывания
        mov     SI, offset signature
        sub     SI, offset ROUT        ;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                ;оставляет нужное количество памяти
        int     21h
;*****
*****
LOADED:
;смотрим, есть ли в хвосте /un , тогда нужно выгрузить
        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    PRINT
    ret
;*****
*****
UNLOAD:
    pop     AX
    pop     ES
    call    DELETE
    mov     dx, offset interrupt_was_unloaded
    call    PRINT
    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 освобожденного
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    PRINT
        pop     DX
        ret
MY_INT ENDP
;-----
;-----
MAIN:
        mov     AX, DATA
        mov     DS, AX
        mov     keep_psp, ES
        call    CHECK_INT
        xor     AL, AL
        mov     AH, 4Ch
        int     21h
CODE ENDS
        END     MAIN

```