

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

Прокопенко Н.

Преподаватель

Ефремов М.А.

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.

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 списка 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.

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.

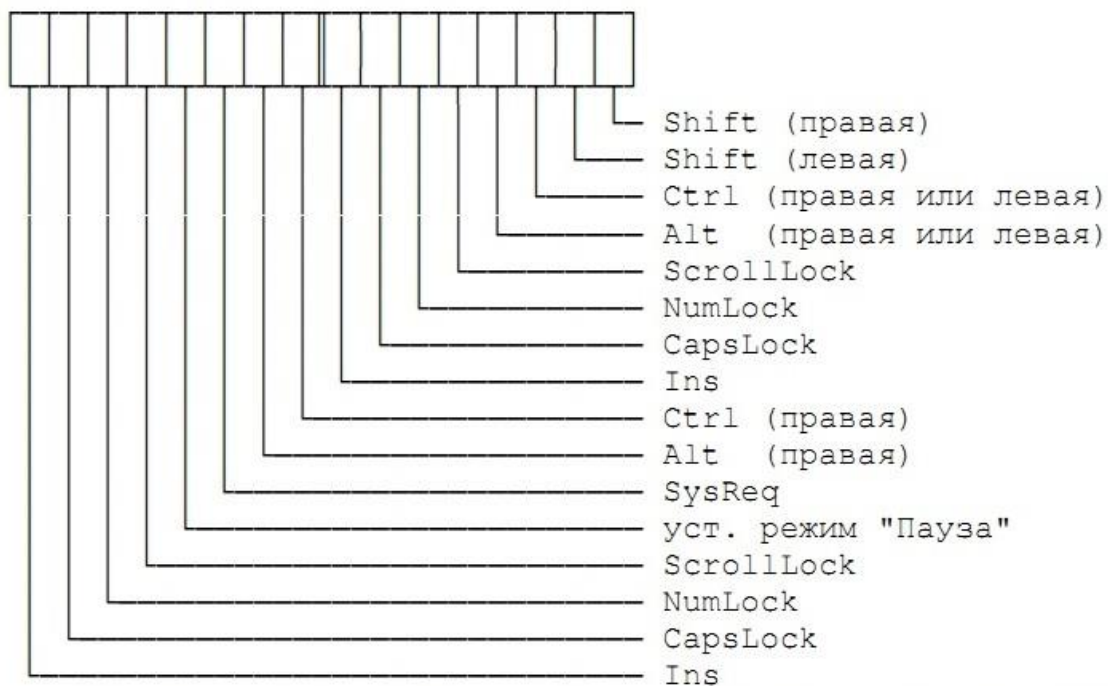
В прерывании клавиатуры можно выделить три основных шага:

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 микро процессору 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 о ять 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 составляются элементами 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 A 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. ^вые 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 отработки 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 отработки аппаратного прерывания
in al,61H ;взять значение порта управления клавиатурой

```

```

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

```

Описание процедур и структур данных представлены в табл. 1 и табл.2 соответственно.

Таблица 1 – описание процедур

Название процедуры	Назначение
INTERRUPT	Функция обработчика прерывания
LAST_BYTE	Устанавливает резидентную функцию для обработки прерывания.
ISLOADED	Проверка установки резидента
CHECK_UNLOAD_FLAG	Проверка команды '/un'
LOAD	Загрузка резидента
UNLOAD	Выгрузка резидента

Таблица 2 – описание структур данных

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

Примеры работы программы по шагам представлены на рисунке 1, рисунке 2, рисунке 3, рисунке 4, рисунке 5, рисунке 6.

```

C:\>lab3_1.com
Amount of available memory: 648912 byte
Extended memory size: 15360 kbyte
Chain of MBC
Address Type MCB Address PSP Size 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 загрузки резидента (используем модуль, разработанный в третьей лабораторной работе)


```

C:\>lab5.exe
Resident loaded

C:\>lab3_1.com
Amount of available memory: 647936 byte
Extended memory size: 15360 kbyte
Chain of MBC
Address Type MCB Address PSP Size SD/SC
016F 4D 0008 16
0171 4D 0000 64
0176 4D 0040 256
0187 4D 0192 144
0191 4D 0192 800 LAB5
01C4 4D 01CF 144
01CE 5A 01CF 647936 LAB3_1

```

Рисунок 2 – загрузка резидента в память с последующей проверкой состояния памяти

```

C:\>lab4_.exe
Resident loaded

C:\>lab4_.exe
Resident already loaded

```

Рисунок 3 – попытка повторной загрузки резидента в память

```

C:\>lab5.exe
Resident loaded

C:\>HelloWorld

```

Рисунок 4 – пример работы программы после загрузки резидента

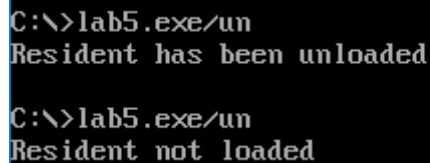
```

C:\>lab5.exe/un
Resident has been unloaded

C:\>lab3_1.com
Amount of available memory: 648912 byte
Extended memory size: 15360 kbyte
Chain of MBC
Address Type MCB Address PSP Size 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 5 – запуск отложенной программы с ключом /un, тем самым выгружаем резидент и состояние памяти после выгрузки резидента



```
C:\>lab5.exe/un  
Resident has been unloaded  
  
C:\>lab5.exe/un  
Resident not loaded
```

Рисунок 6 – попытка повторного запуска отложенной программы
с ключом /un

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.

Код программы lab5_.ASM представлен в приложении А.

Ответы на контрольные вопросы.

- Какого типа прерывания использовались в работе?

В работе использовались прерывания BIOS 9h,16h и пользовательское прерывание DOS 21h.

- 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

LAB5_.ASM

```
ASSUME CS:CODE, DS:DATA, SS:AStack
```

```
AStack SEGMENT STACK
```

```
    DW 64 DUP(?)
```

```
AStack ENDS
```

```
CODE SEGMENT
```

```
INTERRUPT PROC FAR
```

```
    jmp function
```

```
;DATA
```

```
    AD_PSP      dw 0
```

```
    SR_PSP dw 0
```

```
    keep_cs dw 0
```

```
    keep_ip dw 0
```

```
    is_loaded dw 0FFDAh
```

```
    scan_code db 2h, 3h, 4h, 5h, 6h, 7h, 8h, 9h, 0Ah, 0Bh, 82h, 83h,  
84h, 85h, 86h, 87h, 88h, 89h, 8Ah, 8Bh, 00h
```

```
    newtable db ' Helloworld'
```

```
    ss_keeper dw 0
```

```
    sp_keeper dw 0
```

```
    ax_keeper dw 0
```

```
    inter_stack dw 64 dup (?)
```

```
function:
```

```
    mov ss_keeper, ss
```

```
    mov sp_keeper, sp
```

```
    mov ax_keeper, ax
```

```
    mov ax, seg inter_stack
```

```
    mov ss, ax
```

```
    mov sp, 0
```

```
    mov ax, ax_keeper
```

```
    push bx
```

```
    push cx
```

```
    push dx
```

```
    push ax
```

```
    ;Считывание номера клавиши
```

```
    sub ax, ax
```

```
    in al, 60h
```

```
    ;Проверка на требуемые скан-коды
```

```

push ds
push ax
mov ax, SEG scan_code
mov ds, ax
pop ax
mov dx, offset scan_code
;dx - смещение символов, al - сам символ
push bx
push cx
mov bx, dx
sub ah, ah

;Сравнение кодов
for_compare:
mov cl, byte ptr [bx]
cmp cl, 0h
je end_compare
cmp al, cl
jne no_equally
;Совпадает
mov ah, 01h
no_equally:
;Не совпадает
inc bx
jmp for_compare
end_compare:
pop cx
pop bx
pop ds

cmp ah, 01h
je processing
jmp not_processing

not_processing:
;Возврат к стандартному обработчику прерывания
pop ax
mov ss, ss_keeper
mov sp, sp_keeper
pushf
push keep_cs
push keep_ip
iret

```

```

processing:
;Обработка прерывания
push bx
push cx
push dx

cmp al,80h
ja go

push es
push ds
push ax

mov ax, seg newtable
mov ds, ax
mov bx, offset newtable
pop ax

xlatb
pop ds
write_to_buffer:
;Запись в буфер клавиатуры
mov ah, 05h
mov cl, al
sub ch, ch
int 16h
or al, al
jnz cleaning
pop es
go:
jmp @ret
;Очистка буфера и повторение
cleaning:
push ax
mov ax, 40h
mov es, ax
mov word ptr es:[1Ah], 001Eh
mov word ptr es:[1Ch], 001Eh
pop ax
jmp write_to_buffer
@ret:
;Отработка аппаратного прерывания
in al, 61h
mov ah, al

```

```

or al, 80h
out 61h, al
xchg ah, al
out 61h, al
mov al, 20h
out 20h, al

;Восстановление регистров
pop dx
pop cx
pop bx

pop ax
mov ax, ss_keeper
mov ss, ax
mov ax, ax_keeper
mov sp, sp_keeper
;pop ax ;восстановление ax
iret
INTERRUPT ENDP

LAST_BYTE PROC
LAST_BYTE ENDP

ISLOADED PROC near
push dx
push es
push bx

mov ax, 3509h ;получение вектора прерываний
int 21h

mov dx, es:[bx+11]
cmp dx, 0FFDAh ;проверка на совпадение кода
je int_is_loaded
mov al, 0h
pop bx
pop es
pop dx
ret
int_is_loaded:
mov al, 01h
pop bx
pop es

```

```

    pop dx
    ret
ISLOADED ENDP

CHECK_UNLOAD_FLAG PROC near
    push es
    mov ax,AD_PSP
    mov es,ax
    xor bx,bx
    inc bx

    mov al,es:[81h+bx]
    inc bx
    cmp al,'/'
    jne unload_end

    mov al,es:[81h+bx]
    inc bx
    cmp al,'u'
    jne unload_end

    mov al,es:[81h+bx]
    inc bx
    cmp al,'n'
    jne unload_end

    mov al,1h

unload_end:
    pop es
    ret
CHECK_UNLOAD_FLAG ENDP

LOAD PROC near
    push ax
    push bx
    push dx
    push es

    mov ax,3509h
    int 21h
    mov keep_ip,bx
    mov keep_cs,es

```



```

push ds
mov dx,offset INTERRUPT
mov ax,seg INTERRUPT
mov ds,ax
mov ax,2509h
int 21h
pop ds

mov dx,offset int_loaded
mov ah,09h
int 21h

pop es
pop dx
pop bx
pop ax
ret
LOAD ENDP

UNLOAD PROC near
push ax
push bx
push dx
push es

mov ax,3509h
int 21h

cli
push ds
mov dx,es:[bx+9]    ;IP стандартного
mov ax,es:[bx+7]    ;CS стандартного
mov ds,ax
mov ax,2509h
int 21h
pop ds
sti

mov dx,offset int_unload    ;сообщение о выгрузке
mov ah,09h
int 21h

;Удаление MCB

```

```

push es

mov cx,es:[bx+3]
mov es,cx
mov ah,49h
int 21h

pop es
mov cx,es:[bx+5]
mov es,cx
int 21h

pop es
pop dx
pop bx
pop ax
ret
UNLOAD ENDP

Main PROC far

mov bx,02Ch
mov ax,[bx]
mov SR_PSP,ax
mov AD_PSP,ds ;сохраняем PSP
sub ax,ax
xor bx,bx

mov ax,data
mov ds,ax
call CHECK_UNLOAD_FLAG ;Загрузка или выгрузка(проверка параметра)
cmp al,1h
je un_load

call ISLOADED ;Установлен ли разработанный вектор прерывания
cmp al,01h
jne al_loaded

mov dx,offset int_al_loaded ;Уже установлен(выход с сообщение)
mov ah,09h
int 21h

mov ah,4Ch
int 21h

```

```

al_loaded:

;Загрузка
    call LOAD
;Оставляем обработчик прерываний в памяти
    mov dx,offset LAST_BYTE
    mov cl,4h
    shr dx,cl
    inc dx
    add dx,1Ah

    mov ax,3100h
    int 21h

;Выгрузка
un_load:

    call ISLOADED
    cmp al,0h
    je not_loaded

    call UNLOAD

    mov ax,4C00h
    int 21h

not_loaded:
    mov dx,offset int_not_loaded      ;Если резидент не установлен, то
нежелательно выгружать стандартный ВП
    mov ah,09h
    int 21h

    mov ax,4C00h
    int 21h

Main ENDP
CODE ENDS

DATA SEGMENT
    int_not_loaded    db 'Resident not loaded',13,10,'$'
    int_al_loaded     db 'Resident already loaded',13,10,'$'
    int_loaded        db 'Resident loaded',13,10,'$'

```

```
int_unload      db 'Resident has been unloaded',13,10,'$'  
DATA ENDS  
END Main
```