**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. 6381 |  | Герасимова Д.В. |
| Преподаватель |  | Губкин А.Ф. |

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.

**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.

Еслипрерывание 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.

**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.

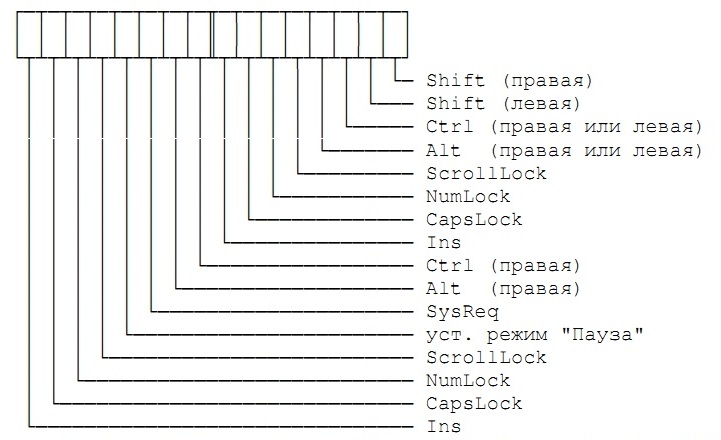
В прерывании клавиатуры можно выделить три основных шага:

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 клавиатуры |
| 0040:001C | 2 | Адрес конца буфера клавиатуры |
| 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 AT, 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 A. 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 со оставляются 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 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 61H,al xchg ah,al

;сохранить его

;установить бит разрешения для клавиатуры

;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

**В программе используются следующие процедуры:**

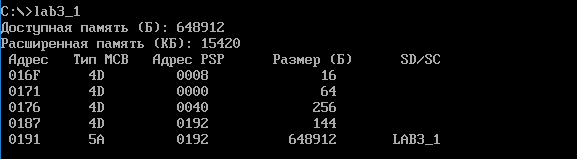
|  |  |
| --- | --- |
| **Название процедуры** | **Назначение** |
| INTERRUPT | Функция обработчика прерывания |
| LAST\_BYTE | Устанавливает резидентную функцию для обработки прерывания. |
| ISLOADED | Проверка установки резидента |
| CHECK\_UNLOAD\_FLAG | Проверка команды '/un' |
| LOAD | Загрузка резидента |
| UNLOAD | Выгрузка резидента |

**Определение структуры 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 |

**Результат работы программ**

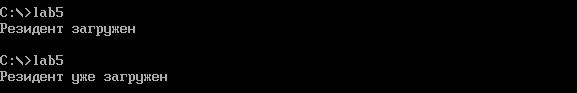
1. Проверим состояние памяти до выполнения разработанного модуля

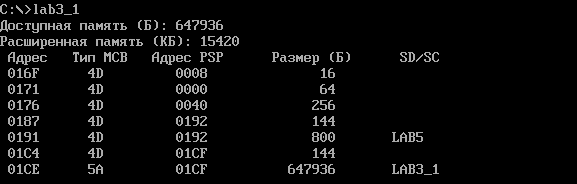


1. Установим резидентный обработчик прерываний



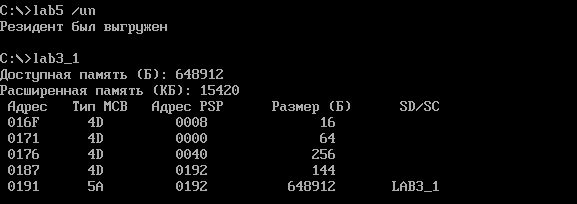
1. Проверим, определяет ли программа установленный обработчик прерываний



1. Проверим состояние памяти на наличие загруженного модуля.
2. Проверим работает ли программа: введем “123456789”



1. Запустим отложенную программу с ключом /un и проверяем состояние памяти после выгрузки резидента



**Ответы на контрольные вопросы:**

1. Какого типа прерывания использовались в работе?

**Ответ:** прерывания функций DOS(21h), прерывания функций BIOS(16h, 09h).

1. Чем отличается скан код от кода ASCII?

**Ответ:** 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.

**Заключение:**

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.

**Приложение**

**Код программы 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 ?

SR\_PSP dw ?

keep\_cs dw ?

keep\_ip dw ?

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 ' abcdefghij'

ss\_keeper dw ?

sp\_keeper dw ?

ax\_keeper dw ?

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

;Востановление 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 'Резидент не загружен',13,10,'$'

int\_al\_loaded db 'Резидент уже загружен',13,10,'$'

int\_loaded db 'Резидент загружен',13,10,'$'

int\_unload db 'Резидент был выгружен',13,10,'$'

DATA ENDS

END Main