**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №4**

**по дисциплине «Операционные системы»**

Тема: Обработка стандартных прерываний

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6381 |  | Лопатина А.С. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Постановка задачи**

**Цель работы.**

В архитектуре компьютера существуют стандартные прерывания, за которыми закреплены определённые вектора прерываний. Вектор прерываний хранит адрес подпрограммы обработчика прерываний. При возникновении прерывания, аппаратура компьютера передаёт управление и выполняет соответствующие действия.

В лабораторной работе номер 4 предлагается построить обработчик прерываний сигналов таймера. Эти сигналы генерируются аппаратурой через определённые интервалы времени и, при возникновении такого сигнала, возникает прерывание с определённым значением вектора. Таким образом, управление будет передано функции, чья точка входа записана в соответствующий вектор прерывания.

**Необходимые сведения для составления программы.**

Резидентные обработчики прерываний - это программные модули, которые вызываются при возникновении прерываний определенного типа (сигнал таймера, нажатие клавиши и т.д.), которым соответствуют определенные вектора прерывания. Когда вызывается прерывание, процессор переключается на выполнение кода обработчика, а затем возвращается на выполнение прерванной программы. Адрес возврата в прерванную программу (CS:IP) запоминается в стеке вместе с регистром флагов. Затем в CS:IP загружается адрес точки входа программы обработки прерывания и начинает выполняться его код. Обработчик прерывания должен заканчиваться инструкцией IRET (возврат из прерывания).

Вектор прерывания имеет длину 4 байта. В первом хранится значение IP, во втором - CS. Младшие 1024 байта памяти содержат 256 векторов. Вектор для прерывания 0 начинается с ячейки 0000:0000, для прерывания 1 - с ячейки 0000:0004 и т.д.

Обработчик прерывание - это отдельная процедура, имеющая следующую структуру:

PROC FAR

PUSH AX ; сохранение изменяемых регистров

<действия по обработке прерывания>

POP AX ; восстановление регистров

MOV AL, 20H

OUT 20H,AL

IRET

ROUT ENDP

Две последние строки необходимы для разрешения обработки прерываний с более низкими уровнями, чем только что обработанное. Для установки написанного прерывания в поле векторов прерываний используется функция 25H прерывания 21H, которая устанавливает вектор прерывания на указанный адрес.

|  |  |  |  |
| --- | --- | --- | --- |
| PUSH | DS |  |  |
| 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, | 1CH | ;номер вектора | |
| INT | 21H |  | ;меняем прерывание | |

Программа, выгружающая обработчик прерываний должна восстанавливать оригинальные векторы прерываний. Функция 35 прерывания 21H позволяет восстановить значение вектора прерывания, помещая значение сегмента в ES, а смещение в BX. Программа должна содержать следующие инструкции:

; -- хранится в обработчике прерываний

KEEP\_CS DW 0 ; для хранения сегмента

KEEP\_IP DW 0 ; и смещения прерывания

; -- в программе при загрузке обработчика прерывания MOV AH, 35H ; функция получения вектора

MOV AL, 1CH ; номер вектора

INT 21H

MOV KEEP\_IP, BX ; запоминание смещения MOV KEEP\_CS, ES ; и сегмента

; -- в программе при выгрузке обработчика прерываний CLI

PUSH DS

MOV DX, KEEP\_IP

MOV AX, KEEP\_CS

MOV DS, AX

MOV AH, 25H

MOV AL, 1CH

INT 21H ; восстанавливаем вектор

POP DS

STI

Для того, чтобы оставить процедуру прерывания резидентной в памяти, следует воспользоваться функцией DOS 31h прерывания 21h. Эта функция оставляет память, размер которой указывается в качестве параметра, занятой, а остальную память освобождает и осуществляет выход в DOS.

Функция 31h int 21h использует следующие параметры:

AH - номер функции 31h;

AL - код завершения программы;

DX - размер памяти в параграфах, требуемый резидентной программе.

Пример обращения к функции: mov DX,offset LAST\_BYTE ; размер в байтах от начала сегмента

mov CL,4 ; перевод в параграфы

shr DX,CL

inc DX ; размер в параграфах

mov AH,31h

int 21h

Вывод на экран информации обработчиком прерываний осуществляется с помощью функций прерывания 10h.

**Сведения о функциях и структурах данных управляющей программы.**

Сведения о функциях:

|  |  |
| --- | --- |
| Название | Назначение |
| OUTPUT\_PROC | Вывод на экран |
| INTERRUPTION | Обработчик прерывания для 1Ch |
| INSTALL\_CHECK | Проверка установки прерывания |
| UN\_CHECK | Проверка на введение /un |
| INSTALL\_INTER | Загрузка обработчика прерывания |
| UNLOAD\_INTER | Выгрузка обработчика прерывания |
| MAIN | Основная функция |

Сведения о структурах данных:

|  |  |  |
| --- | --- | --- |
| Название | Тип | Назначение |
| INSTALL | db | Обработчик прерывания установлен |
| NOT\_INSTALL | db | Обработчик прерывания не установлен |
| ALR\_INSTALL | db | Обработчик прерывания уже установлен |
| UNLOAD | db | Обработчик прерывания выгружен |
| ADDR\_PSP1 | dw | Переменная для хранения PSP |
| ADDR\_PSP2 | dw | Вторая переменная для хранения PSP |
| KEEP\_CS | dw | Переменная для хранения сегмента старого прерывания 1Ch |
| KEEP\_IP | dw | Переменная для хранения смещения старого прерывания 1Ch |
| INTER\_SET | dw | Адрес нового прерывания 1Ch |
| COUNT | db | Счетчик |

**Последовательность действий, выполняемых программой.**

Модуль типа .EXE выполняет следующие функции:

1) Проверяет, установлено ли пользовательское прерывание с вектором 1Ch.

2) Устанавливает резидентную функцию для обработки прерывания настраивает вектор прерываний, если прерывание не установлено, и осуществляется выход о функции 4Ch прерывания int 21h.

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

4) Выгрузка прерывания о 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.

**Результат работы программы.**

1. Запускаем программу и проверяем, установлено ли пользовательское прерывание 1Ch;

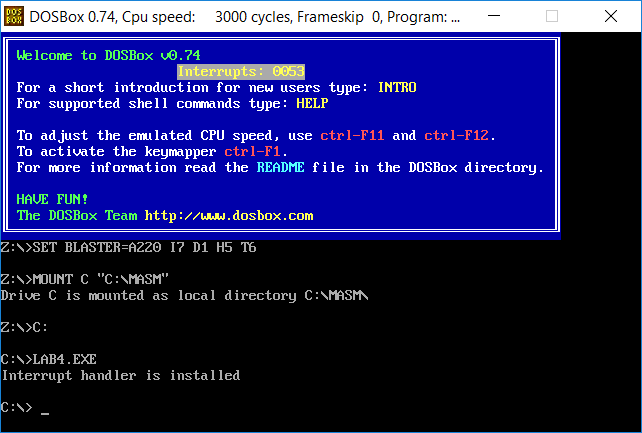


рис.1

1. Проверяем размещение прерывания в памяти. Для этого нужно запускаем программу lab3, которая отображает карту памяти в виде с писка блоков MCB;

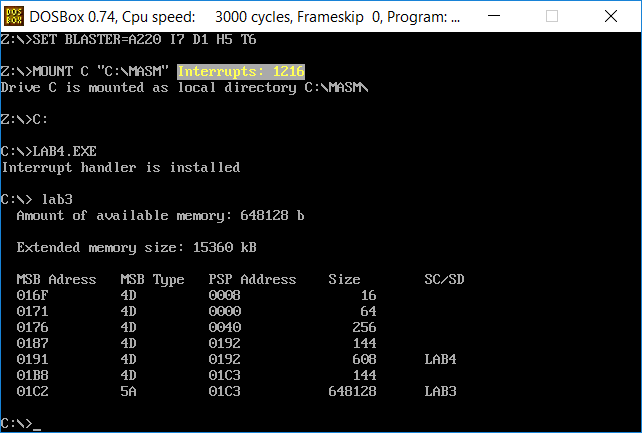


рис.2

1. Запускаем отлаженную программу еще раз и убеждаемся, что программа определяет установленный обработчик прерываний;

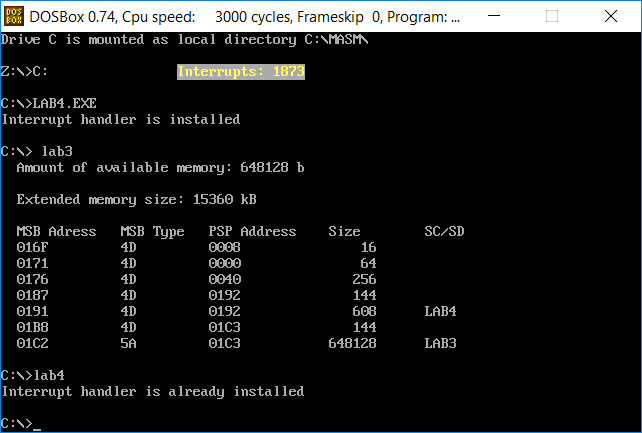


рис.3

1. Запускаем отлаженную программу с ключом выгрузки и убеждаемся, что резидентный обработчик прерывания выгружен;

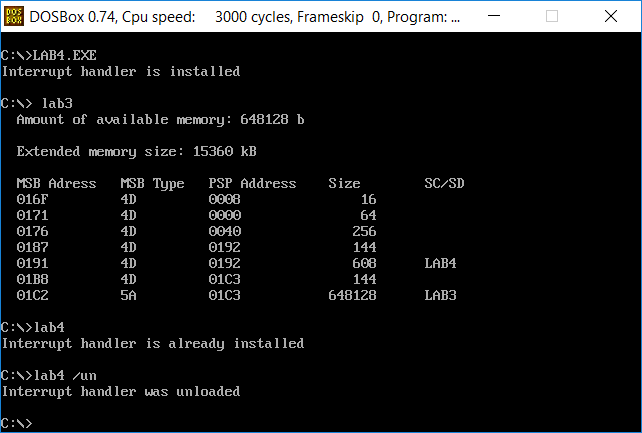


рис.4

1. Убеждаемся, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память, занятая резидентом освобождена. Для этого запускаем программу lab3;

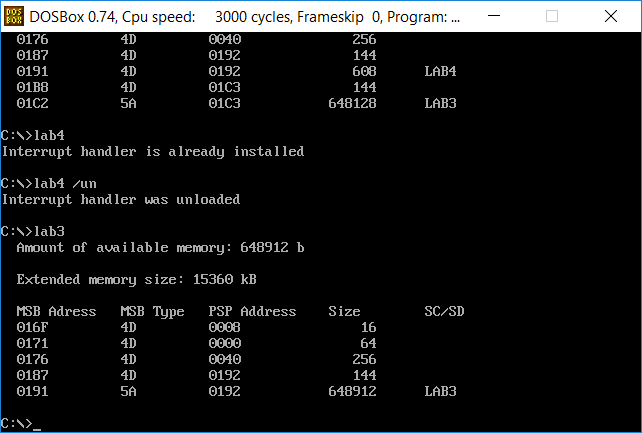


рис.5

**Описание результатов исследования проблем, поставленных в лабораторной работе**

1. Как реализован механизм прерывания от часов?

Ответ: каждые 55 мс:

1. сохраняется содержимое регистров;
2. определяется источник прерывания, по номеру источника прерывания которого определяется смещение в таблице векторов прерываний;
3. сохраняется в CS:IP;
4. управление передается по адресу CS:IP, то есть выполняется запуск обработчика прерывания и происходит его выполнение;
5. происходит возврат управления прерванной программе;
6. Какого типа прерывания использовались в работе?

Ответ: в работе использовались аппаратные прерывания (int 1Сh), прерывания функций DOS (int 21h) и прерывания функций BIOS(int 10h);

**Заключение**

В результате выполнения лабораторной работы бы разработан собственный обработчик прерывания сигналов таймера, изучены дополнительные функции работы с памятью, такие как установка программы-резидента и его выгрузка из памяти.

Приложения

**Код программы** **LAB4**.**asm**

ASSUME CS:CODE, DS:DATA, SS:ASTACK

ASTACK SEGMENT STACK

DW 64 DUP(?)

ASTACK ENDS

CODE SEGMENT

;----------------------------

OUTPUT\_PROC PROC NEAR ;Вывод на экран сообщения

push ax

mov ah, 09h

int 21h

pop ax

ret

OUTPUT\_PROC ENDP

;----------------------------

INTERRUPTION PROC FAR

jmp begin

ADDR\_PSP1 dw 0 ;offset 3

ADDR\_PSP2 dw 0 ;offset 5

KEEP\_CS dw 0 ;offset 7

KEEP\_IP dw 0 ;offset 9

INTER\_SET dw 0ABCDh ;offset 11

COUNT db 'Interrupts: 0000 $' ;offset 13

begin:

push ax

push bx

push cx

push dx

mov ah, 3h ;Cчитывание позиции курсора

mov bh, 0h

int 10h

push dx

mov ah,02h ;Установка курсора в определённую позицию

mov bh,0h

mov dx,216h

int 10h

push si ;Считаем количество прерываний

push cx

push ds

mov ax, SEG COUNT

mov ds, ax

lea si, COUNT

add si, 0Fh

mov ah,[si]

inc ah

mov [si], ah

cmp ah, 3Ah

jne output

mov ah, 30h

mov [si], ah

mov bh, [si - 1]

inc bh

mov [si - 1], bh

cmp bh, 3Ah

jne output

mov bh, 30h

mov [si - 1], bh

mov ch, [si - 2]

inc ch

mov [si - 2], ch

cmp ch, 3Ah

jne output

mov ch, 30h

mov [si - 2], ch

mov dh, [si - 3]

inc dh

mov [si - 3], dh

cmp dh, 3Ah

jne output

mov dh, 30h

mov [si - 3],dh

output:

pop ds

pop cx

pop si

push es ;Вывод на экран

push bp

mov ax, SEG COUNT

mov es, ax

lea ax, COUNT

mov bp, ax

mov ah, 13h

mov al, 00h

mov cx, 10h

mov bh, 0h

int 10h

pop bp

pop es

pop dx ;Возвращаем курсор

mov ah, 02h

mov bh, 0h

int 10h

pop dx

pop cx

pop bx

pop ax

iret

INTERRUPTION ENDP

;----------------------------

inter\_end:

INSTALL\_CHECK PROC NEAR ;Проверка установки прерывания

push bx

push dx

push es

mov ah, 35h ;Получение вектора прерываний

mov al, 1Ch ;Функция выдает значение сегмента в ES, смещение в BX

int 21h

mov dx, es:[bx + 11]

cmp dx, 0ABCDh ;Проверка на совпадение кода прерывания

je install\_

mov al, 00h

jmp end\_install

install\_:

mov al, 01h

jmp end\_install

end\_install:

pop es

pop dx

pop bx

ret

INSTALL\_CHECK ENDP

;----------------------------

UN\_CHECK PROC NEAR ;Проверка на то, не ввёл ли пользователь /un

push es

mov ax, ADDR\_PSP1

mov es, ax

cmp byte ptr es:[82h], '/'

jne not\_enter

cmp byte ptr es:[83h], 'u'

jne not\_enter

cmp byte ptr es:[84h], 'n'

jne not\_enter

mov al, 1h

not\_enter:

pop es

ret

UN\_CHECK ENDP

;----------------------------

INSTALL\_INTER PROC NEAR ;Загрузка обработчика прерывания

push ax

push bx

push dx

push es

mov ah, 35h

mov al, 1Ch

int 21h

mov KEEP\_IP, bx ;Запоминаем смещение и сегмент

mov KEEP\_CS, es

push ds

lea dx, INTERRUPTION

mov ax, seg INTERRUPTION

mov ds, ax

mov ah, 25h

mov al, 1Ch

int 21h

pop ds

lea dx, INSTALL

call OUTPUT\_PROC

pop es

pop dx

pop bx

pop ax

ret

INSTALL\_INTER ENDP

;----------------------------

UNLOAD\_INTER PROC NEAR ;Выгрузка обработчика прерывания

push ax

push bx

push dx

push es

mov ah, 35h

mov al, 1Ch

int 21h

cli

push ds

mov dx, es:[bx + 9]

mov ax, es:[bx + 7]

mov ds, ax

mov ah, 25h

mov al, 1Ch

int 21h

pop ds

sti

lea dx, UNLOAD

call OUTPUT\_PROC

push es ;Удаление MCB

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

mov ah, 4Ch ;Выход из программы через функцию 4C

int 21h

ret

UNLOAD\_INTER ENDP

;----------------------------

MAIN PROC FAR

mov bx,2Ch

mov ax,[bx]

mov ADDR\_PSP2,ax

mov ADDR\_PSP1,ds ;сохраняем PSP

mov dx, ds

sub ax,ax

xor bx,bx

mov ax,data

mov ds,ax

xor dx, dx

call UN\_CHECK ;Проверка на введение /un

cmp al, 01h

je unload\_

call INSTALL\_CHECK ;Проверка не является ли программа резидентной

cmp al, 01h

jne not\_resident

lea dx, ALR\_INSTALL ;Программа уже загружена

call OUTPUT\_PROC

jmp quit

;Загрузка резидента

not\_resident:

call INSTALL\_INTER

lea dx, inter\_end

mov cl, 04h

shr dx, cl

add dx, 1Bh

mov ax, 3100h

int 21h

;Выгрузка резидента

unload\_:

call INSTALL\_CHECK

cmp al, 0h

je not\_install\_

call UNLOAD\_INTER

jmp quit

;Прерывание выгружено

not\_install\_:

lea dx, UNLOAD

call OUTPUT\_PROC

quit:

mov ah, 4Ch

int 21h

MAIN ENDP

CODE ENDS

DATA SEGMENT

INSTALL db 'Interrupt handler is installed', 0dh, 0ah, '$'

NOT\_INSTALL db 'Interrupt handler is not installed', 0dh, 0ah, '$'

ALR\_INSTALL db 'Interrupt handler is already installed', 0dh, 0ah, '$'

UNLOAD db 'Interrupt handler was unloaded', 0dh, 0ah, '$'

DATA ENDS

END Main