**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

|  |  |  |
| --- | --- | --- |
| Студент гр. 6383 |  | Никитин К.В. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**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 по соответствующему адресу вектора прерывания. Обработчик прерываний получает управление и выполняет соответствующие действия.

В лабораторной работе № 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.

**Описание функций.**

|  |  |
| --- | --- |
| **Название функции** | **Назначение** |
| ROUT | пользовательский обработчик прерываний, считающий и печатающий количество его вызовов |
| setCurs | устанавливает курсор в строку dh, колонку dl |
| getCurs | возвращает положение курсора в dh, dl |
| outputBP | функция вывода строки по адресу ES:BP |
| PRINT | вызывает функцию печати строки |
| PROV\_ROUT | проверяет, установлен ли пользовательский обработчик прерывания, и если нет – устанавливает его. В ином случае, если хвост равен ‘/un’, восстанавливает стандартное |
| SET\_ROUT | устанавливает пользовательское прерывание |
| DEL\_ROUT | удаляет пользовательское прерывание |
| SAVE\_STAND | сохраняет адрес стандартного прерывания в KEEP\_IP, KEEP\_CS |
| BYTE\_TO\_HEX | переводит число AL в коды символов 16-ой с/с, записывая получившееся в al и ah |
| TETR\_TO\_HEX | вспомогательная функция для работы функции BYTE\_TO\_HEX |
| WRD\_TO\_HEX | переводит число AX в строку в 16-ой с/с, записывая получившееся в di, начиная с младшей цифры |

**Последовательность действий, выполняемых утилитой.**

1. Состояние памяти до запуска lab4.exe представлено на Рис.1 (использовалась программа 1.com):

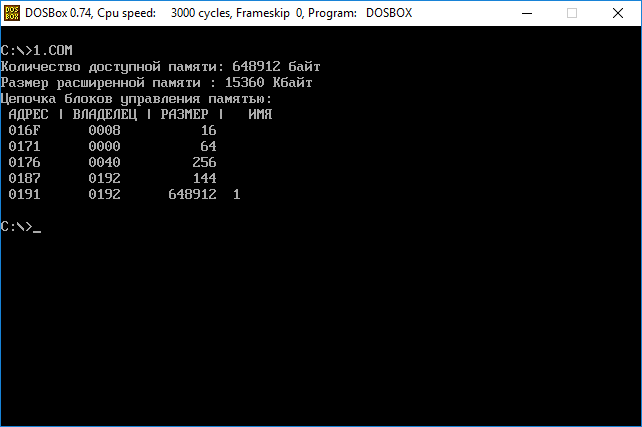


Рисунок 1 – Результат работы программы 1.com

1. Запуск программы lab4.exe представлен на Рис.2:

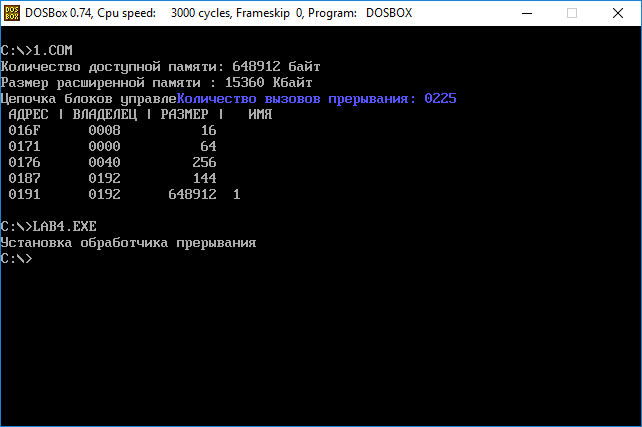


Рисунок 2 – Результат работы программы lab4.exe

1. Проверим размещение прерывания в памяти с помощью программы 1.com, которая отображает карту памяти в виде списка блоков MCB(см. Рис.3):

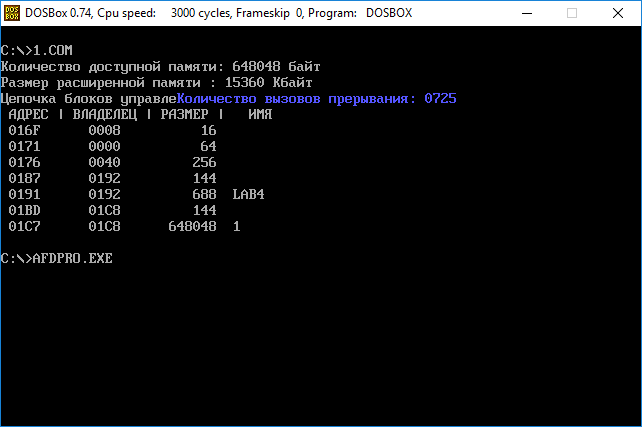


Рисунок 3 – Результат работы программы 1.com после запуска lab4.exe

1. Запустим программу lab4.exe повторно (см. Рис.4):

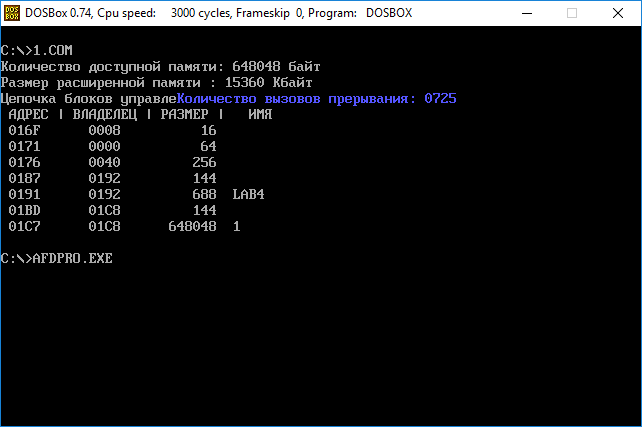


Рисунок 4 – Результат повтороного запуска программы lab4.exe

1. Запустим программу lab4.exe с ключом выгрузки /un (см. Рис.5):

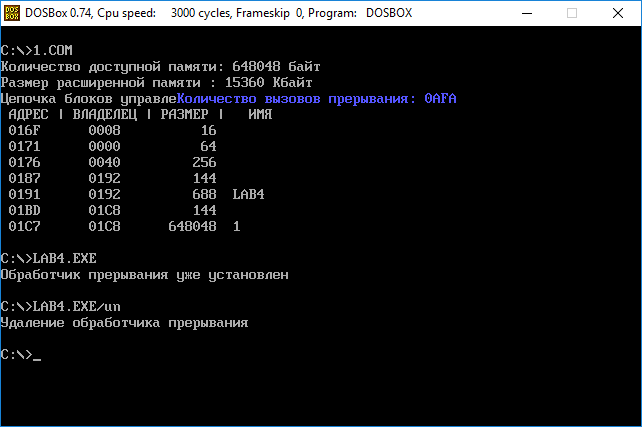


Рисунок 5 – Результат запуска программы lab4.exe c ключом /un

1. Убедимся, что память освобождена, используя программу 1.com (см. Рис.6):

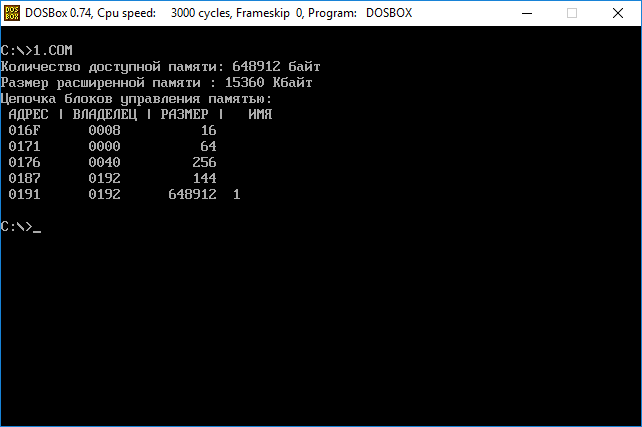


Рисунок 6 – Результат выполнения программы 1.com

**Ответы на контрольные вопросы.**

1. Как реализован механизм прерывания от часов?

Сначала сохраняется содержимое регистров, потомопределяется источник прерывания, по номеру которого определяется смещение в таблице векторов прерывания, сохраняется в CS : IP, передаётся управление по адресу CS:IP и происходит выполнение обработчика, и в конце происходит возврат управления прерванной программе. Прерывания генерируются системным таймером с частотой 18,206 Гц.

1. Какого типа прерывания использовались в работе?

В работе использовались аппаратные прерывания(int 1Ch) и программные(int 21h, int 10h).

**Вывод.**

В процессе выполнения данной лабораторной работы был построен обработчик прерываний сигналов таймера.

**ПРИЛОЖЕНИЕ А**

CODE SEGMENT

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

ROUT PROC FAR

jmp go\_

SIGNATURA dw 0ABCDh

KEEP\_PSP dw 0

KEEP\_IP dw 0

KEEP\_CS dw 0

INT\_STACK DW 100 dup (?)

COUNT dw 0

KEEP\_SS DW 0

KEEP\_AX DW ?

KEEP\_SP DW 0

MSG db 'Количество вызовов прерывания: $'

go\_:

mov KEEP\_SS, SS

mov KEEP\_SP, SP

mov KEEP\_AX, AX

mov AX,seg INT\_STACK

mov SS,AX

mov SP,0

mov AX,KEEP\_AX

push ax

push bp

push es

push ds

push dx

push di

mov ax,cs

mov ds,ax

mov es,ax

mov ax,CS:COUNT

add ax,1

mov CS:COUNT,ax

mov di,offset MSG+34

call WRD\_TO\_HEX

mov bp,offset MSG

call outputBP

pop di

pop dx

pop ds

pop es

pop bp

mov al,20h

out 20h,al

pop ax

mov AX,KEEP\_SS

mov SS,AX

mov AX,KEEP\_AX

mov SP,KEEP\_SP

iret

ROUT ENDP

TETR\_TO\_HEX PROC near

and AL,0Fh

cmp AL,09

jbe NEXT

add AL,07

NEXT: add AL,30h

ret

TETR\_TO\_HEX ENDP

BYTE\_TO\_HEX PROC near

push CX

mov AH,AL

call TETR\_TO\_HEX

xchg AL,AH

mov CL,4

shr AL,CL

call TETR\_TO\_HEX

pop CX

ret

BYTE\_TO\_HEX ENDP

WRD\_TO\_HEX PROC near

push BX

mov BH,AH

call BYTE\_TO\_HEX

mov [DI],AH

dec DI

mov [DI],AL

dec DI

mov AL,BH

call BYTE\_TO\_HEX

mov [DI],AH

dec DI

mov [DI],AL

pop BX

ret

WRD\_TO\_HEX ENDP

outputBP PROC near

push ax

push bx

push dx

push cx

mov ah,13h

mov al,0

mov bl,09h

mov bh,0

mov dh,4

mov dl,22

mov cx,35

int 10h

pop cx

pop dx

pop bx

pop ax

ret

outputBP ENDP

LAST\_BYTE:

PRINT PROC

push ax

mov ah,09h

int 21h

pop ax

ret

PRINT ENDP

PROV\_ROUT PROC

mov ah,35h

mov al,1ch

int 21h

mov si,offset SIGNATURA

sub si,offset ROUT

mov ax,0ABCDh

cmp ax,ES:[BX+SI]

je ROUT\_EST

call SET\_ROUT

jmp PROV\_KONEC

ROUT\_EST:

call DEL\_ROUT

PROV\_KONEC:

ret

PROV\_ROUT ENDP

SET\_ROUT PROC

mov ax,KEEP\_PSP

mov es,ax

cmp byte ptr es:[80h],0

je UST

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

jne UST

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

jne UST

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

jne UST

mov dx,offset PRER\_NE\_SET\_MSG

call PRINT

ret

UST:

call SAVE\_STAND

mov dx,offset PRER\_SET\_MSG

call PRINT

push ds

mov dx,offset ROUT

mov ax,seg ROUT

mov ds,ax

mov ah,25h

mov al,1ch

int 21h

pop ds

mov dx,offset LAST\_BYTE

mov cl,4

shr dx,cl

add dx,1

add dx,20h

xor AL,AL

mov ah,31h

int 21h

xor AL,AL

mov AH,4Ch

int 21H

SET\_ROUT ENDP

DEL\_ROUT PROC

push dx

push ax

push ds

push es

mov ax,KEEP\_PSP

mov es,ax

cmp byte ptr es:[80h],0

je UDAL\_KONEC

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

jne UDAL\_KONEC

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

jne UDAL\_KONEC

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

jne UDAL\_KONEC

mov dx,offset PRER\_DEL\_MSG

call PRINT

mov ah,35h

mov al,1ch

int 21h

mov si,offset KEEP\_IP

sub si,offset ROUT

mov dx,es:[bx+si]

mov ax,es:[bx+si+2]

mov ds,ax

mov ah,25h

mov al,1ch

int 21h

mov ax,es:[bx+si-2]

mov es,ax

mov ax,es:[2ch]

push es

mov es,ax

mov ah,49h

int 21h

pop es

mov ah,49h

int 21h

jmp UDAL\_KONEC2

UDAL\_KONEC:

mov dx,offset PRER\_UZHE\_SET\_MSG

call PRINT

UDAL\_KONEC2:

pop es

pop ds

pop ax

pop dx

ret

DEL\_ROUT ENDP

SAVE\_STAND PROC

push ax

push bx

push es

mov ah,35h

mov al,1ch

int 21h

mov KEEP\_CS, ES

mov KEEP\_IP, BX

pop es

pop bx

pop ax

ret

SAVE\_STAND ENDP

BEGIN:

mov ax,DATA

mov ds,ax

mov KEEP\_PSP, es

call PROV\_ROUT

xor AL,AL

mov AH,4Ch

int 21H

CODE ENDS

DATA SEGMENT

PRER\_SET\_MSG db 'Установка обработчика прерывания','$'

PRER\_DEL\_MSG db 'Удаление обработчика прерывания',0DH,0AH,'$'

PRER\_UZHE\_SET\_MSG db 'Обработчик прерывания уже установлен',0DH,0AH,'$'

PRER\_NE\_SET\_MSG db 'Обработчик прерывания не установлен',0DH,0AH,'$'

STRENDL db 0DH,0AH,'$'

DATA ENDS

ASTACK SEGMENT STACK

dw 100h dup (?)

ASTACK ENDS

END BEGIN