Министерство образования Республики Беларусь

Учреждение образования Белорусский государственный университет информатики и радиоэлектроники

Кафедра ЭВМ

Отчёт по лабораторной работе №2

“Программирование контроллера прерываний”

Проверил: Выполнил:

к.т.н., доцент студент гр.150502

Одинец Дмитрий Николаевич Альхимович Н. Г.

Минск 2023

**Задача**

Целью данной работы является написание резидентной программы, выполняющей перенос всех векторов аппаратных прерываний ведущего и ведомого контроллеров на пользовательский прерывания.

Требуется, чтобы программа выполняла следующее:

1. Вывод на экран в двоичной форме следующие регистры контроллеров

прерывания (как ведущего, так и ведомого):

* регистр запросов на прерывания;
* регистр обслуживаемых прерываний;
* регистр масок.

2. Осуществлять переход на стандартные обработчики аппаратных прерываний, для обеспечения нормальной работы компьютера.

**Алгоритм**

Программа состоит из нескольких частей программы, представляющих собой некоторые функции, с помощью которых осуществляется следующее:

1. Все векторы аппаратных прерываний ведущего и ведомого контроллера

переносятся на пользовательские прерывания с помощью функций getvect и setvect.

1. Инициализация контроллеров с применением последовательности

команд: ICW1, ICW2, ICW3 и ICW4.

1. С помощью функции \_dos\_keep осуществляется выход в DOS, при этом

программа остаётся резидентной.

1. В каждом обработчике выводятся в видеопамять в двоичной форме

значения регистров запросов на прерывания, обслуживаемых прерываний, масок. Затем вызываются стандартные обработчики прерываний.

**Листинг программы**

#include <stdio.h>

#include <stdlib.h>

#include <dos.h>

#include <conio.h>

struct VIDEO

{

unsigned char sym;

unsigned char attribute;

};

int colour = 0xf0;

void interrupt(\*system\_8)(...); //указатель на прерывание

void interrupt(\*system\_9)(...);

void interrupt(\*system\_10)(...);

void interrupt(\*system\_11)(...);

void interrupt(\*system\_12)(...);

void interrupt(\*system\_13)(...);

void interrupt(\*system\_14)(...);

void interrupt(\*system\_15)(...);

void interrupt(\*system\_70)(...);

void interrupt(\*system\_71)(...);

void interrupt(\*system\_72)(...);

void interrupt(\*system\_73)(...);

void interrupt(\*system\_74)(...);

void interrupt(\*system\_75)(...);

void interrupt(\*system\_76)(...);

void interrupt(\*system\_77)(...);

//VIDEO far\* screen = (VIDEO far\*)MK\_FP(0xB800, 0);

void print()

{

char temp;

int i, value;

VIDEO far\* screen = (VIDEO far\*)MK\_FP(0xB800, 0);

value = inp(0x21); //регистр масок ведущего контроллера

for(i=0; i<8; i++)

{

temp = value % 2;

value >>= 1;

screen->sym = temp + '0';

screen->attribute = colour;

screen++;

}

screen++;

value = inp(0xA1); //регистр масок ведомого контроллера

for(i=0; i<8; i++)

{

temp = value % 2;

value >>= 1;

screen->sym = temp + '0';

screen->attribute = colour;

screen++;

}

screen += 63;

outp(0x20, 0x0A);

value = inp(0x20); //регистр запросов на прерывания ведущего контроллера

for(i=0; i<8; i++)

{

temp = value % 2;

value >>= 1;

screen->sym = temp + '0';

screen->attribute = colour;

screen++;

}

screen++;

outp(0xA0, 0x0A);

value = inp(0xA0); //регистр запросов на прерывания ведомого контроллера

for(i=0; i<8; i++)

{

temp = value % 2;

value >>= 1;

screen->sym = temp + '0';

screen->attribute = colour;

screen++;

}

screen += 63;

outp(0x20, 0x0B);

value = inp(0x20); //регистр обслуживаемых прерываний ведущего контроллера

for(i=0; i<8; i++)

{

temp = value % 2;

value >>= 1;

screen->sym = temp + '0';

screen->attribute = colour;

screen++;

}

screen++;

outp(0xA0, 0x0B);

value = inp(0xA0); //регистр обслуживаемых прерываний ведомого контроллера

for(i=0; i<8; i++)

{

temp = value % 2;

value >>= 1;

screen->sym = temp + '0';

screen->attribute = colour;

screen++;

}

}

void interrupt custom\_08(...) //прототип функции обработки прерывания

{

print();

(\*system\_8)();

}

void interrupt custom\_09(...)

{

print();

(\*system\_9)();

}

void interrupt custom\_10(...)

{

print();

(\*system\_10)();

}

void interrupt custom\_11(...)

{

print();

(\*system\_11)();

}

void interrupt custom\_12(...)

{

print();

(\*system\_12)();

}

void interrupt custom\_13(...)

{

print();

(\*system\_13)();

}

void interrupt custom\_14(...)

{

print();

(\*system\_14)();

}

void interrupt custom\_15(...)

{

print();

(\*system\_15)();

}

void interrupt custom\_60(...)

{

print();

(\*system\_70)();

}

void interrupt custom\_61(...)

{

print();

(\*system\_71)();

}

void interrupt custom\_62(...)

{

print();

(\*system\_72)();

}

void interrupt custom\_63(...)

{

print();

(\*system\_73)();

}

void interrupt custom\_64(...)

{

print();

(\*system\_74)();

}

void interrupt custom\_65(...)

{

print();

(\*system\_75)();

}

void interrupt custom\_66(...)

{

print();

(\*system\_76)();

}

void interrupt custom\_67(...)

{

print();

(\*system\_77)();

}

int main()

{

unsigned far\* fp;

//изменяем таблицу векторов прерывания

//IRQ0-7

system\_8 = getvect(0x8); //IRQ0 прерывание таймера, возникает 18,2 раза в секунду

system\_9 = getvect(0x9); //IRQ1 прерывание от клавиатуры

system\_10 = getvect(0xA); //IRQ2 используется для каскадирования аппаратных прерываний (ведомое)

system\_11 = getvect(0xB); //IRQ3 прерывание асинхронного порта COM2

system\_12 = getvect(0xC); //IRQ4 прерывание асинхронного порта COM1

system\_13 = getvect(0xD); //IRQ5 прерывание от контроллера жесткого диска для XT

system\_14 = getvect(0xE); //IRQ6 прерывание генерируется контроллером флоппи диска

system\_15 = getvect(0xF); //IRQ7 прерывание принтера

//IRQ8-15

system\_70 = getvect(0x70); //IRQ8 прерывание от часов реального времени

system\_71 = getvect(0x71); //IRQ9 прерывание от контроллера EGA

system\_72 = getvect(0x72); //IRQ10 зарезервировано

system\_73 = getvect(0x73); //IRQ11 зарезервировано

system\_74 = getvect(0x74); //IRQ12 зарезервировано

system\_75 = getvect(0x75); //IRQ13 прерывание от математического сопроцессора

system\_76 = getvect(0x76); //IRQ14 прерывание от контроллера жесткого диска

system\_77 = getvect(0x77); //IRQ15 зарезервировано

setvect(0x08, custom\_60);

setvect(0x09, custom\_61);

setvect(0x0A, custom\_62);

setvect(0x0B, custom\_63);

setvect(0x0C, custom\_64);

setvect(0x0D, custom\_65);

setvect(0x0E, custom\_66);

setvect(0x0F, custom\_67);

setvect(0x60, custom\_08);

setvect(0x61, custom\_09);

setvect(0x62, custom\_10);

setvect(0x63, custom\_11);

setvect(0x64, custom\_12);

setvect(0x65, custom\_13);

setvect(0x66, custom\_14);

setvect(0x67, custom\_15);

\_disable(); //запретить прерывания (= cli)

//ведущий контроллер

outp(0x20, 0x11); //ICW1, размер вектора прерывания - 8 байт

outp(0x21, 0x08); //ICW2 - 78

outp(0x21, 0x04); //ICW3

outp(0x21, 0x01); //ICW4

//ведомый контроллер

outp(0xA0, 0x11); //ICW1

outp(0xA1, 0x08); //ICW2

outp(0xA1, 0x02); //ICW3

outp(0xA1, 0x01); //ICW4

\_enable(); //разрешить прерывания (= sti)

system("cls");

FP\_SEG(fp) = \_psp; //возвращает сегмент far-указателя

FP\_OFF(fp) = 0x2c; //возвращает смещение сегмента данных с переменными среды

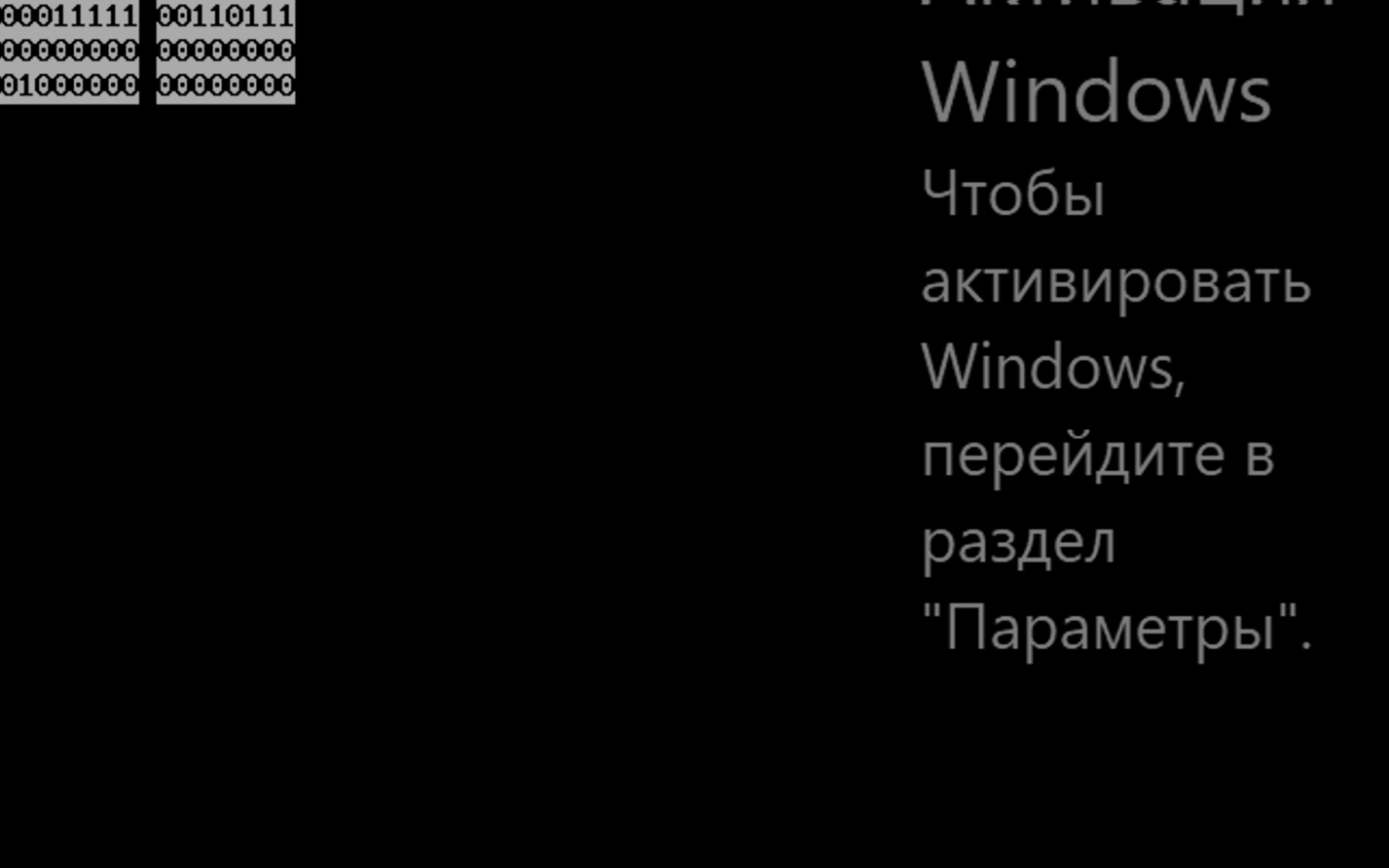
\_dos\_freemem(\*fp); //освобождает блок памяти, первый байт которого расположен в fp

\_dos\_keep(0, (\_DS - \_CS) + (\_SP / 16) + 1); //оставить программу резидентной (первый параметр - код завершения, второй - объем памяти, который должен быть зарезервирован для программы после ее завершения

return 0;

}

**Тест**

****

**Заключение**

В данной лабораторной работе разработана программа, которая переносит вектора аппаратных прерываний ведущего и ведомого контроллера на вектора пользовательских прерываний.

Для компиляции программы использовался TurboC++, а для запуска исполняемого файла – DOSBox 0.74-3.