Министерство образования Республики Беларусь

Учреждение образования Белорусский государственный университет информатики и радиоэлектроники

Кафедра ЭВМ

Отчёт по лабораторной работе №3

“Программирование контроллера прерываний”

Проверил: Выполнил:

к.т.н., доцент студент гр.950501

Одинец Дмитрий Николаевич Поддубный Д.П.

Минск 2020

**Задача**

Написать резидентную программу выполняющую перенос всех векторов аппаратных

прерываний ведущего и ведомого контроллера на пользовательские прерывания. При этом

необходимо написать обработчики аппаратных прерываний, которые будут установлены на

используемые пользовательские прерывания и будут выполнять следующие функции:

1 Выводить на экран в двоичной форме следующие регистры контроллеров

прерывания (как ведущего, так и ведомого):

регистр запросов на прерывания;

регистр обслуживаемых прерываний;

регистр масок.

При этом значения регистров должны выводиться всегда в одно и то же место экрана.

2 Осуществлять переход на стандартные обработчики аппаратных прерываний, для

обеспечения нормальной работы компьютера.

**Алгоритм**

Для реализации данной программы следует переопределить регистры, а после вернуть их в исходное состояние.

**Листинг программы**

#include<dos.h>

#include<stdio.h>

#include<stdlib.h>

void printToVideoMemory(char\* str, int x, int y, unsigned char attribute);

void byteToString(unsigned char temp, char \*str);

void print();

void interrupt new\_8(...);

void interrupt new\_9(...);

void interrupt new\_10(...);

void interrupt new\_11(...);

void interrupt new\_12(...);

void interrupt new\_13(...);

void interrupt new\_14(...);

void interrupt new\_15(...);

void interrupt new\_70(...);

void interrupt new\_71(...);

void interrupt new\_72(...);

void interrupt new\_73(...);

void interrupt new\_74(...);

void interrupt new\_75(...);

void interrupt new\_76(...);

void interrupt new\_77(...);

void interrupt (\*old\_8)(...);

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

void main()

{

// изменяем таблицу векторов прерывания

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

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

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

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

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

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

old\_14 = getvect(0xE); // IRQ6 прерывание генерируется контроллером флоппи диска после завершения операции

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

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

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

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

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

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

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

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

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

setvect(0x08, new\_8);

setvect(0x09, new\_9);

setvect(0x0A, new\_10);

setvect(0x0B, new\_11);

setvect(0x0C, new\_12);

setvect(0x0D, new\_13);

setvect(0x0E, new\_14);

setvect(0x0F, new\_15);

setvect(0x78, new\_70);

setvect(0x79, new\_71);

setvect(0x7A, new\_72);

setvect(0x7B, new\_73);

setvect(0x7C, new\_74);

setvect(0x7D, new\_75);

setvect(0x7E, new\_76);

setvect(0x7F, new\_77);

// ИНИЦИАЛИЗАЦИЯ

// Ведущий контроллер

unsigned char value = inp(0x21); // Запоминаем регистр масок

outp(0x20, 0x11); // ICW1 (инициализация ведущего контроллера) - ICW4 будет вызвана,

// размер вектора прерывания - 8 байт

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

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

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

outp(0x21, value); // Восстанавливаем регистр масок

// Ведомый контроллер

value = inp(0xA1); // Запоминаем регистр масок

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

outp(0xA1, 0x78); // ICW2

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

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

outp(0xa1, value); // Восстанавливаем регистр масок

\_dos\_keep(0,(\_DS-\_CS)+(\_SP/16)+1); // Оставить программу резидентной

}

// Байт в строку в двоичном виде

void byteToString(unsigned char temp, char \*str)

{

int i;

str[8] = 0;

i=7;

while(temp)

{

str[i]='0'+temp%2;

temp=temp/2;

i--;

}

for(;i>-1;i--)

str[i]='0';

}

void printToVideoMemory(char\* str, int x, int y, unsigned char attribute)

{

char far\* start = (char far\*)0xb8000000;

start += x+160\*y;

int i = 0;

while(str[i] != 0)

{

\*start = str[i];

start++;

\*start = attribute;

start++;

i++;

}

}

void print()

{

unsigned char isr\_master, isr\_slave; // Interrupt Service Register - Регистр обслуживаемых прерываний

unsigned char irr\_master, irr\_slave; // Interrupt Request Register - Регистр запросов на прерывания

unsigned char imr\_master, imr\_slave; // Interrupt Mask Register - Регистр масок

imr\_master = inp(0x21);

imr\_slave = inp(0xA1);

outp(0x20, 0x0A);

irr\_master = inp(0x20);

outp(0x20, 0x0B);

isr\_master = inp(0x20);

outp(0xA0,0x0A);

irr\_slave = inp(0xA0);

outp(0xA0,0x0B);

isr\_slave = inp(0xA0);

char str[9];

printToVideoMemory("MASTER PIC ->> ISR: ",0, 0, 0x6E);

byteToString(isr\_master, str);

printToVideoMemory(str, 44, 0, 0x6E);

printToVideoMemory(" | IRR: ",60, 0, 0x6E);

byteToString(irr\_master, str);

printToVideoMemory(str, 80, 0, 0x6E);

printToVideoMemory(" | IMR: ", 96, 0, 0x6E);

byteToString(imr\_master, str);

printToVideoMemory(str, 116, 0, 0x6E);

printToVideoMemory("SLAVE PIC ->> ISR: ", 0, 1, 0x1E);

byteToString(isr\_slave, str);

printToVideoMemory(str, 44, 1, 0x1E);

printToVideoMemory(" | IRR: ", 60, 1, 0x1E);

byteToString(irr\_slave, str);

printToVideoMemory(str, 80, 1, 0x1E);

printToVideoMemory(" | IMR: ",96, 1, 0x1E);

byteToString(imr\_slave, str);

printToVideoMemory(str, 116, 1, 0x1E);

}

void interrupt new\_8(...)

{

print();

(\*old\_8)();

}

void interrupt new\_9(...)

{

print();

(\*old\_9)();

}

void interrupt new\_10(...)

{

print();

(\*old\_10)();

}

void interrupt new\_11(...)

{

print();

(\*old\_11)();

}

void interrupt new\_12(...)

{

print();

(\*old\_12)();

}

void interrupt new\_13(...)

{

print();

(\*old\_13)();

}

void interrupt new\_14(...)

{

print();

(\*old\_14)();

}

void interrupt new\_15(...)

{

print();

(\*old\_15)();

}

void interrupt new\_70(...)

{

print();

(\*old\_70)();

}

void interrupt new\_71(...)

{

print();

(\*old\_71)();

}

void interrupt new\_72(...)

{

print();

(\*old\_72)();

}

void interrupt new\_73(...)

{

print();

(\*old\_73)();

}

void interrupt new\_74(...)

{

print();

(\*old\_74)();

}

void interrupt new\_75(...)

{

print();

(\*old\_75)();

}

void interrupt new\_76(...)

{

print();

(\*old\_76)();

}

void interrupt new\_77(...)

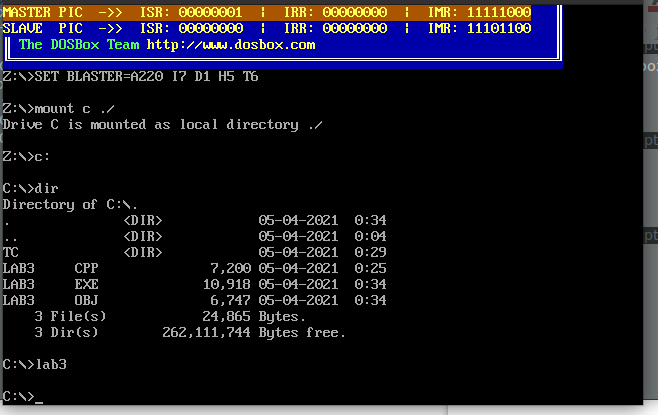
{

print();

(\*old\_77)();

}

**Тест**



**Заключение**

В ходе данной работы была изучена работа с резистентными программами на примере программы, которая переопределяет аппартные прерывания.