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.250502

Одинец Дмитрий Николаевич Бригадир А. С.

Минск 2024

**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**

Программа состоит из нескольких подпрограмм (частей программы), представляющих собой некоторые функции. К ним относятся функции:

* Вывод состояния регистра
* Общий вывод информации
* Новое определение каждого прерывания
* Переопределение прерываний
* Создание резидентной программы

**Листинг программы**

#include <dos.h>

#include <time.h>

#include <stdio.h>

#include <stdlib.h>

void printRegister(int reg, char far \*pos)

{

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

{

\*pos = reg % 2 + '0';

pos += 2;

reg = reg >> 1;

}

}

void print()

{

char temp;

int i, val;

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

screen += 20;

val = inp(0x21); // get mask Master registor

printRegister(val, screen);

screen += 18;

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

printRegister(val, screen);

screen += 142;

outp(0x20, 0x0A);

val = inp(0x20);

printRegister(val, screen); // get Masters's request register

screen += 18;

outp(0xA0, 0x0A);

val = inp(0xA0); // get Slave's request register

printRegister(val, screen);

screen += 142;

outp(0x20, 0x0B);

val = inp(0x20); // get Master's service register

printRegister(val, screen);

screen += 18;

outp(0xA0, 0x0B);

val = inp(0xA0); // get Slave's service register

printRegister(val, screen);

}

void prepare()

{

\_disable();

print();

}

void interrupt (\*oldint1[8])(...); // IRQ 0-7

void interrupt (\*oldint2[8])(...); // IRQ 8-15

void interrupt newint80(...)

{

\_disable();

print();

oldint1[0]();

\_enable();

}

void interrupt newint81(...)

{

prepare();

oldint1[1]();

\_enable();

}

void interrupt newint82(...)

{

prepare();

oldint1[2]();

\_enable();

}

void interrupt newint83(...)

{

prepare();

oldint1[3]();

\_enable();

}

void interrupt newint84(...)

{

prepare();

oldint1[4]();

\_enable();

}

void interrupt newint85(...)

{

prepare();

oldint1[5]();

\_enable();

}

void interrupt newint86(...)

{

prepare();

oldint1[6]();

\_enable();

}

void interrupt newint87(...)

{

prepare();

oldint1[7]();

\_enable();

}

void interrupt newint08(...)

{

prepare();

oldint2[0]();

\_enable();

}

void interrupt newint09(...)

{

prepare();

oldint2[1]();

\_enable();

}

void interrupt newint0A(...)

{

prepare();

oldint2[2]();

\_enable();

}

void interrupt newint0B(...)

{

prepare();

oldint2[3]();

\_enable();

}

void interrupt newint0C(...)

{

prepare();

oldint2[4]();

\_enable();

}

void interrupt newint0D(...)

{

prepare();

oldint2[5]();

\_enable();

}

void interrupt newint0E(...)

{

prepare();

oldint2[6]();

\_enable();

}

void interrupt newint0F(...)

{

prepare();

oldint2[7]();

\_enable();

}

void initialize()

{

int i;

for (i = 0x70; i <= 0x77; i++)

{

oldint2[i - 0x70] = getvect(i);

}

for (i = 0x08; i <= 0x0F; i++)

{

oldint1[i - 0x08] = getvect(i);

}

// set new handlers for IRQ 0-7

setvect(0x80, newint80);

setvect(0x81, newint81);

setvect(0x82, newint82);

setvect(0x83, newint83);

setvect(0x84, newint84);

setvect(0x85, newint85);

setvect(0x86, newint86);

setvect(0x87, newint87);

// set new handlers for IRQ8-15

setvect(0x08, newint08);

setvect(0x09, newint09);

setvect(0x0A, newint0A);

setvect(0x0B, newint0B);

setvect(0x0C, newint0C);

setvect(0x0D, newint0D);

setvect(0x0E, newint0E);

setvect(0x0F, newint0F);

\_disable(); // CLI

// interrupt initializtion for Master

outp(0x20, 0x11); // ICW1 - initialize master (00010001, 0001 - base, last 1 - next will be)

outp(0x21, 0x80); // ICW2 - base vector for master (10101000, )

outp(0x21, 0x04); // ICW3 - the port bit of Slave (in binary format)

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

// interrupt initialization for Slave

outp(0xA0, 0x11); // ICW1 - initialize Slave

outp(0xA1, 0x08); // ICW2 - base vector for slave

outp(0xA1, 0x02); // ICW3 - the port number of connected port on Master

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

\_enable(); // STI

}

int main()

{

unsigned far \*fp;

initialize();

system("cls");

printf("mask: \n");

printf("obsl:\n");

printf("requ:\n");

FP\_SEG(fp) = \_psp;

FP\_OFF(fp) = 0x2c;

\_dos\_freemem(\*fp);

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

return 0;

}

**Тест**

****

**Заключение**

В данной лабораторной работе разработана резидентная программа, выполняющая перенос всех векторов аппаратных прерываний ведущего и ведомого контроллера на пользовательские прерывания.