БГУИР

Кафедра ЭВМ

Отчет по лабораторной работе №3

Тема: «Подсистема прерываний»

Выполнил:

студент группы 750501 Новицкий А.Д.

Проверил:

к.т.н., доцент Одинец Д.Н.

Минск

2019

**СОДЕРЖАНИЕ**

1. Постановка задач………………………………………………………………..3

2. Алгоритм решения задачи………………………………………………….......4

3. Листинг программы…………………………………………………………..5-8

4. Тестовые примеры……………………………………………………………...9

5. Заключение…………………………………………………………………….10

**ПОСТАНОВКА ЗАДАЧ**

Под MS DOS написать программу, которая:

1) выполняет инициализацию контроллера прерываний;

2) выводит на экран содержимое регистров запросов, обслуживаний и масок для ведущего и ведомого контроллеров (через видеобуфер).

\*При нажатии на клавиши меняется цвет или фон выводимой информации.

Программа должна быть резидентной. Все вектора прерываний переопределяются, новый базовый адрес выбирается в соответствии с вариантом.

**АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ**

1. Получить старые обработчики прерываний по базовым адресам
2. Установить новые обработчики включающие функции вывода на экран состояний регистров и старые обработчики по новым адресам
3. Запретить прерывания
4. Установить новые базовые адреса для контроллера прерываний
5. Разрешить прерывания
6. Получить сегмент и смещение сегмента данных
7. Освободить память
8. Оставить программу резидентной

**ЛИСТИНГ ПРОГРАММЫ**

#include <stdio.h>

#include <stdlib.h>

#include<dos.h>

char color = 0x00;

struct VIDEO

{

unsigned char symb;

unsigned char attr;

};

void get\_reg();

void print(int, int, int);

void printstr(int, int, int);

void interrupt(\*int8) (...);

void interrupt(\*int9) (...);

void interrupt(\*intA) (...);

void interrupt(\*intB) (...);

void interrupt(\*intC) (...);

void interrupt(\*intD) (...);

void interrupt(\*intE) (...);

void interrupt(\*intF) (...);

void interrupt(\*int88) (...);

void interrupt(\*int89) (...);

void interrupt(\*int8A) (...);

void interrupt(\*int8B) (...);

void interrupt(\*int8C) (...);

void interrupt(\*int8D) (...);

void interrupt(\*int8E) (...);

void interrupt(\*int8F) (...);

void interrupt new8(...) { get\_reg(); int8(); }

void interrupt new9(...) { color++; get\_reg(); int9(); }

void interrupt newA(...) { get\_reg(); intA(); }

void interrupt newB(...) { get\_reg(); intB(); }

void interrupt newC(...) { get\_reg(); intC(); }

void interrupt newD(...) { get\_reg(); intD(); }

void interrupt newE(...) { get\_reg(); intE(); }

void interrupt newF(...) { get\_reg(); intF(); }

void interrupt new88(...) { get\_reg(); int88(); }

void interrupt new89(...) { get\_reg(); int89(); }

void interrupt new8A(...) { get\_reg(); int8A(); }

void interrupt new8B(...) { get\_reg(); int8B(); }

void interrupt new8C(...) { color++; get\_reg(); int8C(); }

void interrupt new8D(...) { get\_reg(); int8D(); }

void interrupt new8E(...) { get\_reg(); int8E(); }

void interrupt new8F(...) { get\_reg(); int8F(); }

void print(int val, int y, int x)

{

char temp;

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

screen += (y \* 80 + x);

int i;

for (i = 7; i >= 0; i--) {

temp = val & 1;

val /= 2;

screen->symb = temp + '0';

color &= 15;

screen->attr = color;

screen++;

}

}

void printstr(int num, int y, int x)

{

char str1[] = { 'M', 'A', 'S', 'T', 'E', 'R', ' ', ' ', ' ', 'I', 'R', 'R', '\0' };

char str2[] = { 'I', 'S', 'R', '\0' };

char str3[] = { 'I', 'M', 'R', '\0' };

char str4[] = { 'S', 'L', 'A', 'V', 'E', ' ', ' ', ' ', ' ', 'I', 'R', 'R', '\0' };

int i;

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

screen += (y \* 80 + x);

switch (num)

{

case 1:

{

for (i = 0; str1[i] != 0; i++)

{

screen->symb = str1[i];

screen->attr = 0x04;

screen++;

}

break;

}

case 2:

{

for (i = 0; str2[i] != 0; i++)

{

screen->symb = str2[i];

screen->attr = 0x04;

screen++;

}

break;

}

case 3:

{

for (i = 0; str3[i] != 0; i++)

{

screen->symb = str3[i];

screen->attr = 0x04;

screen++;

}

break;

}

case 4:

{

for (i = 0; str4[i] != 0; i++)

{

screen->symb = str4[i];

screen->attr = 0x04;

screen++;

}

break;

}

}

}

void get\_reg()

{

printstr(1, 13, 25);

outp(0x20, 0x0A);

print(inp(0x20), 14, 34);

printstr(2, 15, 34);

outp(0x20, 0x0B);

print(inp(0x20), 16, 34);

printstr(3, 17, 34);

print(inp(0x21), 18, 34);

printstr(4, 19, 25);

outp(0xA0, 0x0A);

print(inp(0xA0), 20, 34);

printstr(2, 21, 34);

outp(0xA0, 0x0B);

print(inp(0xA0), 22, 34);

printstr(3, 23, 34);

print(inp(0xA1), 24, 34);

}

void init() {

int8 = getvect(0x08);

int9 = getvect(0x09);

intA = getvect(0x0A);

intB = getvect(0x0B);

intC = getvect(0x0C);

intD = getvect(0x0D);

intE = getvect(0x0E);

intF = getvect(0x0F);

int88 = getvect(0x70);

int89 = getvect(0x71);

int8A = getvect(0x72);

int8B = getvect(0x73);

int8C = getvect(0x74);

int8D = getvect(0x75);

int8E = getvect(0x76);

int8F = getvect(0x77);

setvect(0x08, new8);

setvect(0x09, new9);

setvect(0x0A, newA);

setvect(0x0B, newB);

setvect(0x0C, newC);

setvect(0x0D, newD);

setvect(0x0E, newE);

setvect(0x0F, newF);

setvect(0x88, new88);

setvect(0x89, new89);

setvect(0x8A, new8A);

setvect(0x8B, new8B);

setvect(0x8C, new8C);

setvect(0x8D, new8D);

setvect(0x8E, new8E);

setvect(0x8F, new8F);

\_disable();

outp(0x20, 0x11);

outp(0x21, 0x08);

outp(0x21, 0x04);

outp(0x21, 0x05);

outp(0xA0, 0x11);

outp(0xA1, 0x88);

outp(0xA1, 0x02);

outp(0xA1, 0x01);

\_enable();

}

int main()

{

unsigned far \*fp;

init();

FP\_SEG(fp) = \_psp;

FP\_OFF(fp) = 0x2c;

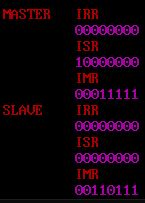
\_dos\_freemem(\*fp);

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

return 0;

}

**ТЕСТОВЫЕ ПРИМЕРЫ**

****

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения лабораторной работы были получены базовые навыки работы с контроллером прерываний и таблицей векторов прерываний.

Контроллер прерываний – это специализированное устройство, служащее для обработки поступающих запросов на прерывания в соответствии с их приоритетом и установленными масками.

Вектор прерывания – это номер и адрес обработчика прерывания.

Таблица векторов прерываний – специализированное место в оперативной памяти занимающее 1 КБ связывающее номер прерывания с адресом его обработчика.

Вложенное прерывание – это когда поступает прерывание с большим приоритетом чем текущее и оно должно быть выполнено до завершения текущего.

Команда IRET служить для возврата управления из прерывания в прерванную программу.

Команда EOI необходима для сброса бита прерывания, который блокирует прерывания с таким же и более низким приоритетом, для того чтобы дать возможность выполняться другим прерываниям.

Команды CLI и SLI запрещают и разрешают внешние прерывания.