Министерство образования Республики Беларусь

Учреждение образования Белорусский государственный университет

информатики и радиоэлектроники

# Кафедра ЭВМ

### Отчет по лабораторной работе № 3

"Программирование контроллера прерываний"

Вариант 7

Выполнил:

студент группы 950502

Киреев Ю.В.

Проверил:

Одинец Д. Н.

#### Минск 2021

1. **Задание**

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.

1. Осуществлять переход на стандартные обработчики аппаратных прерываний, для обеспечения нормальной работы компьютера.
2. **Теоретические сведения**

Каждый момент времени центральный процессор может работать только с одним устройством. Циклический опрос каждого устройства с последующей обработкой запроса оказался неэффективным. Решение задачи оказался контроллер прерываний, который принимает запросы от устройств и в соответствии с приоритетом направляет их процессору, если прерывание от данного устройства не замаскировано (разрешено) в регистре масок.

Если прерывание разрешено и устройство его запросило, то устанавливается соответствующий устройству бит в регистре запросов. Контроллер прерываний состоит из двух микросхем, подключенных каскадно (ведущий и ведомый контроллеры), каждая из которых имеет по 8 линий прерываний (IRQ0-IRQ7, IRQ8- IRQ15). За каждой линией закреплено определенное устройство. Когда процессор получает запрос на прерывание, он сохраняет свое текущее состояние и переключается на выполнение запрошенной операции. При этом устанавливается бит в регистре обслуживания (бит запроса сбрасывается). После обслуживания прерываний сбрасывается бит обслуживания, посылается сигнал EOI (endofinterrupt), процессор переключается на выполняемую ранее задачу.

Для доступа к контроллеру прерываний используются порты 20h и 21h (для ведущего), A0 иA1h (для ведомого). Регистр масок доступен через порт 21h /A1h. Чтобы изменить определенный бит, нужно считать значение из этого регистра, изменить нужный бит, записать значение обратно. Чтобы считать регистр запросов, его нужно выбрать (записать в 20h/A0h значение 0Ah), а затем считать содержимое из порта 20h/A0h. Чтобы считать регистр обслуживания, его нужно выбрать (записать в 20h/A0h значение 0Bh), а затем считать содержимое из порта 20h/A0h.

Для резидентной программы понадобится следующий фрагмент кода:

unsigned far \*fp; //объявляем указатель

FP\_SEG (fp) = \_psp; // получаем сегмент

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

\_dos\_freemem(\*fp); //чтобы его освободить

\_dos\_keep(0,(\_DS -\_CS)+(\_SP/16)+1);//оставляем резидентной, указывая первым параметром код завершения, а вторым - объем памяти, который должен быть зарезервирован для программы после ее завершения

1. **Листинг программы**

#include <dos.h>

#include <time.h>

#include <stdio.h>

#include <stdlib.h>

#define COLORS 7

struct VIDEO

{

unsigned char symbol;

unsigned char attribute;

};

// array of counts for video buffer:

unsigned char colors[COLORS] = { 0x71, 0x62, 0x43, 0x54, 0x35, 0x26, 0x17 };

char color = 0x89;

void changeColor()

{

color = colors[rand() % COLORS];

return;

}

void print()

{

char temp;

int i, val;

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

val = inp(0x21);

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

{

temp = val % 2;

val = val >> 1;

screen->symbol = temp + '0';

screen->attribute = color;

screen++;

}

screen++;

val = inp(0xA1);

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

{

temp = val % 2; // get last bit

val = val >> 1; // move bits on right

screen->symbol = temp + '0';

screen->attribute = color;

screen++; // get to next symbol

}

screen += 63;

outp(0x20, 0x0A);

val = inp(0x20);

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

{

temp = val % 2;

val = val >> 1;

screen->symbol = temp + '0'; // save bit as a char ('0' or '1')

screen->attribute = color;// set color of print symbol by video card

screen++;

}

screen++;

outp(0xA0, 0x0A);

val = inp(0xA0);

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

{

temp = val % 2;

val = val >> 1;

screen->symbol = temp + '0';

screen->attribute = color;

screen++;

}

screen += 63;

outp(0x20, 0x0B);

val = inp(0x20);

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

{

temp = val % 2;

val = val >> 1;

screen->symbol = temp + '0';

screen->attribute = color;

screen++;

}

screen++;

outp(0xA0, 0x0B);

val = inp(0xA0);

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

{

temp = val % 2;

val = val >> 1;

screen->symbol = temp + '0';

screen->attribute = color;

screen++;

void interrupt(\*oldint8) (...);

void interrupt(\*oldint9) (...);

void interrupt(\*oldint10) (...);

void interrupt(\*oldint11) (...);

void interrupt(\*oldint12) (...);

void interrupt(\*oldint13) (...);

void interrupt(\*oldint14) (...);

void interrupt(\*oldint15) (...);

void interrupt(\*oldint70) (...);

void interrupt(\*oldint71) (...);

void interrupt(\*oldint72) (...);

void interrupt(\*oldint73) (...);

void interrupt(\*oldint74) (...);

void interrupt(\*oldint75) (...);

void interrupt(\*oldint76) (...);

void interrupt(\*oldint77) (...);

void interrupt newint08(...) { print(); oldint8(); }

void interrupt newint09(...) { changeColor(); print(); oldint9(); }

void interrupt newint0A(...) { changeColor(); print(); oldint10(); }

void interrupt newint0B(...) { changeColor(); print(); oldint11(); }

void interrupt newint0C(...) { changeColor(); print(); oldint12(); }

void interrupt newint0D(...) { changeColor(); print(); oldint13(); }

void interrupt newint0E(...) { changeColor(); print(); oldint14(); }

void interrupt newint0F(...) { changeColor(); print(); oldint15(); }

void interrupt newintA0(...) { changeColor(); print(); oldint70(); }

void interrupt newintA1(...) { changeColor(); print(); oldint71(); }

void interrupt newintA2(...) { changeColor(); print(); oldint72(); }

void interrupt newintA3(...) { changeColor(); print(); oldint73(); }

void interrupt newintA4(...) { changeColor(); print(); oldint74(); }

void interrupt newintA5(...) { changeColor(); print(); oldint75(); }

void interrupt newintA6(...) { changeColor(); print(); oldint76(); }

void interrupt newintA7(...) { changeColor(); print(); oldint77(); }

void initialize()

{

//IRQ 0-7

oldint8 = getvect(0x08);

oldint9 = getvect(0x09);

oldint10 = getvect(0x0A);

oldint11 = getvect(0x0B);

oldint12 = getvect(0x0C);

oldint13 = getvect(0x0D);

oldint14 = getvect(0x0E);

oldint15 = getvect(0x0F);

oldint70 = getvect(0x70);

oldint71 = getvect(0x71);

oldint72 = getvect(0x72);

oldint73 = getvect(0x73);

oldint74 = getvect(0x74);

oldint75 = getvect(0x75);

oldint76 = getvect(0x76);

oldint77 = getvect(0x77);

setvect(0xA0, newint08);

setvect(0xA1, newint09);

setvect(0xA2, newint0A);

setvect(0xA3, newint0B);

setvect(0xA4, newint0C);

setvect(0xA5, newint0D);

setvect(0xA6, newint0E);

setvect(0xA7, newint0F);

setvect(0x08, newintA0);

setvect(0x09, newintA1);

setvect(0x0A, newintA2);

setvect(0x0B, newintA3);

setvect(0x0C, newintA4);

setvect(0x0D, newintA5);

setvect(0x0E, newintA6);

setvect(0x0F, newintA7);

\_disable();

outp(0x20, 0x11);

outp(0x21, 0xA0);

outp(0x21, 0x04);

outp(0x21, 0x01);

outp(0xA0, 0x11);

outp(0xA1, 0x08);

outp(0xA1, 0x02);

outp(0xA1, 0x01);

\_enable();

}

int main()

{

unsigned far\* p;

initialize();

system("cls");

printf(" - MASK\n");

printf(" - REQUEST\n");

printf(" - SERVICE\n");

printf(" MASTER SLAVE\n");

FP\_SEG(p) = \_psp;

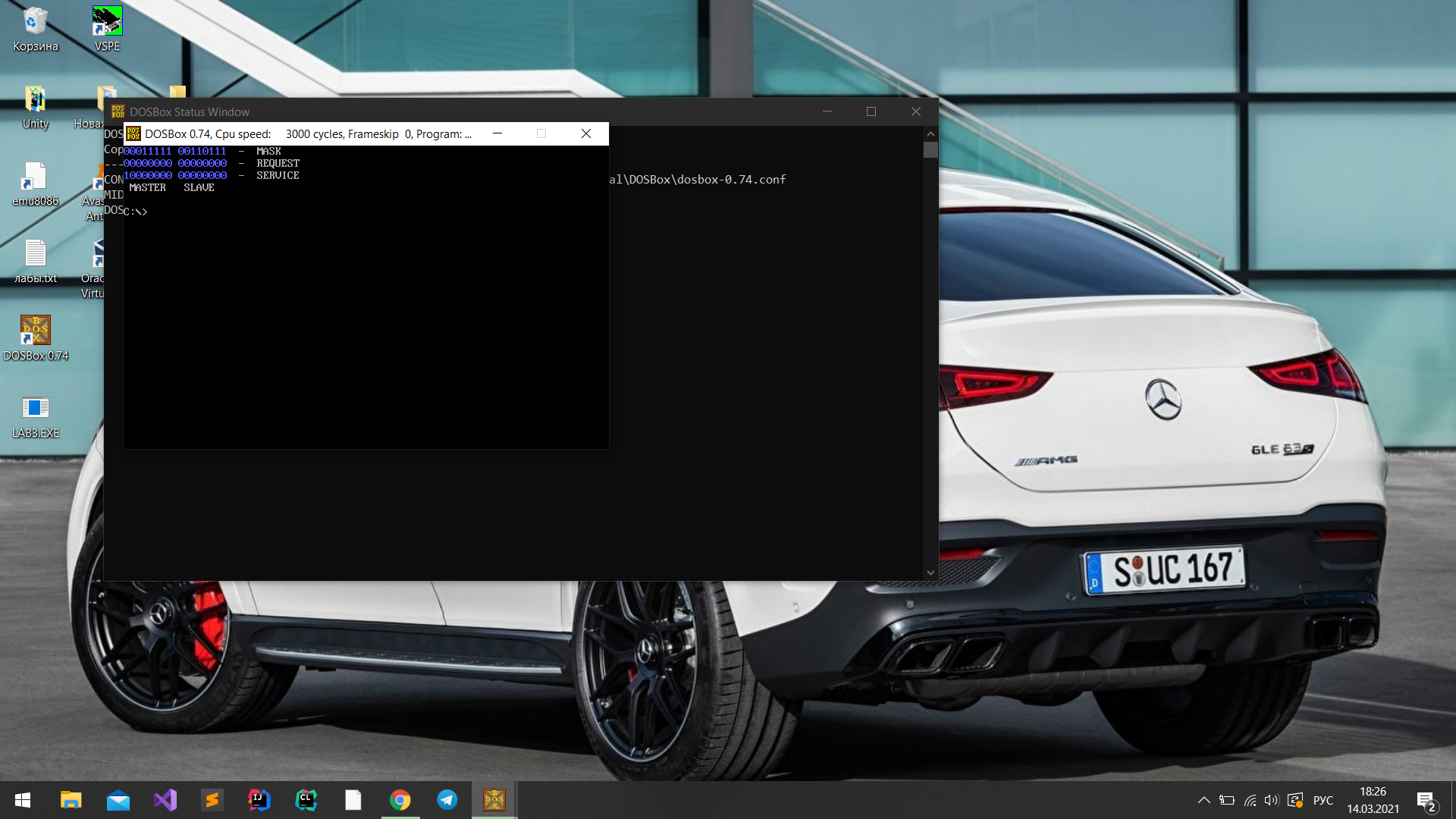
FP\_OFF(p) = 0x2c;

\_dos\_freemem(\*p);

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

return 0;

}



**Вывод:** в процессе лабораторной работы был запрограммирован контроллер прерываний.