Министерство образования Республики Беларусь
Учреждение образования Белорусский государственный университет информатики и радиоэлектроники

Кафедра ЭВМ

Отчёт по лабораторной работе №3
“Программирование контроллера прерываний”

Проверил:
к.т.н., доцент
Одинец Дмитрий Николаевич

Выполнил:
студент гр.950501
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();
```

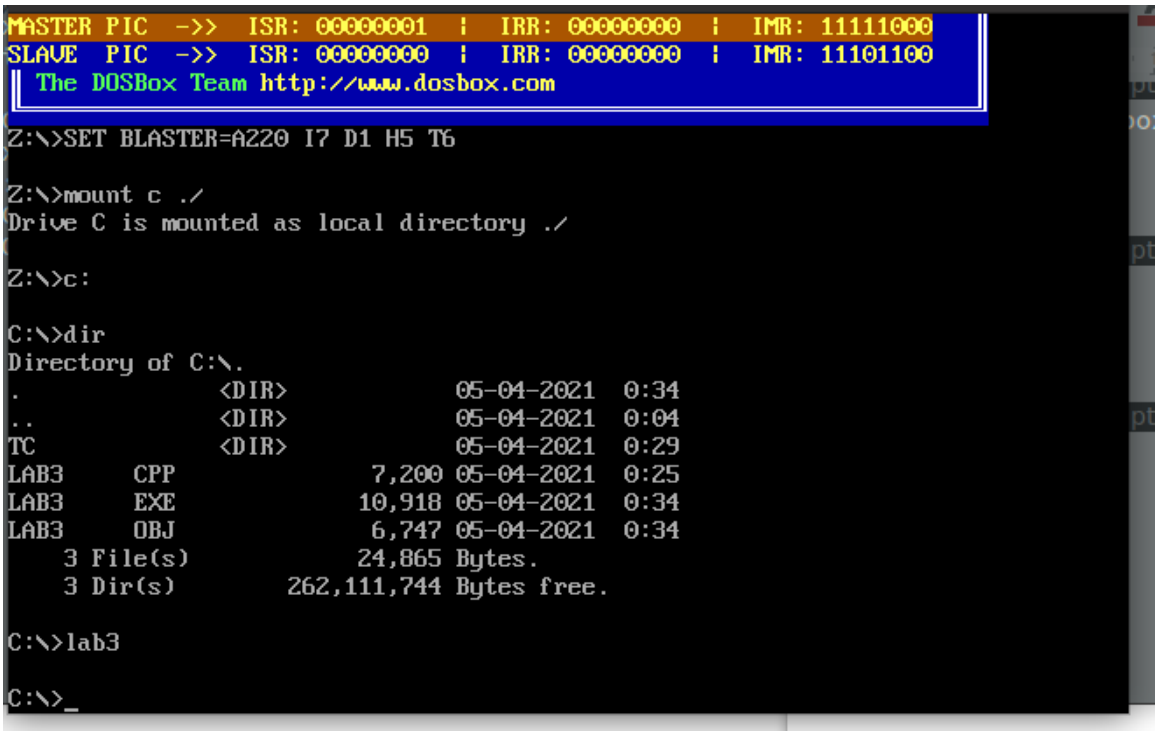
```
(*old_8)();
}

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.