

Министерство образования Республики Беларусь  
Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра ЭВМ

Лабораторная работа № 3  
« Программирование контроллера прерываний»  
Вариант 28

Проверил:  
Одинец Д.Н.  
Выполнил: ст.гр. 950501  
Яценко С. С.

Минск, 2021

## **1. Постановка задачи**

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.

## **2. Алгоритм решения задачи**

1. Запоминаем старые прерывания
2. Устанавливаем новые прерывания согласно варианту
3. Обработка прерываний и вывод регистров
4. Переход на стандартные обработчики прерываний

## 3 Листинг кода программы

### 3.1 Листинг программы на C++

```
#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()
{
    old_8 = getvect(0x8);
    old_9 = getvect(0x9);
    old_10 = getvect(0xA);
    old_11 = getvect(0xB);
    old_12 = getvect(0xC);
    old_13 = getvect(0xD);
    old_14 = getvect(0xE);
    old_15 = getvect(0xF);

    old_70 = getvect(0x70);
    old_71 = getvect(0x71);
    old_72 = getvect(0x72);
    old_73 = getvect(0x73);
    old_74 = getvect(0x74);
    old_75 = getvect(0x75);
    old_76 = getvect(0x76);
    old_77 = getvect(0x77);
    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(0xb8, new_70);
        setvect(0xb9, new_71);
        setvect(0xba, new_72);
        setvect(0xbb, new_73);
        setvect(0xbc, new_74);
        setvect(0xbd, new_75);
        setvect(0xbe, new_76);
        setvect(0xbf, new_77);

        unsigned char value = inp(0x21);
        outp(0x20, 0x11);
        outp(0x21, 0x08);    // ICW2
        outp(0x21, 0x04);    // ICW3
        outp(0x21, 0x01);    // ICW4
        outp(0x21, value);
        value = inp(0xA1);
        outp(0xA0, 0x11);    // ICW1
        outp(0xA1, 0xb8);    // ICW2
        outp(0xA1, 0x02);    // ICW3
        outp(0xA1, 0x01);    // ICW4
        outp(0xA1, value);

        _dos_keep(0, (_DS-_CS)+(_SP/16)+1);
    }

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;
    unsigned char irr_master, irr_slave;
    unsigned char imr_master, imr_slave;

    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) ();
    }

```

## 4 Примеры работы программы

```
DOSBox version 0.74-3
Copyright 2002-2019 DOSBox Team, published under GNU GPL.
---
CONFIG:Loading primary settings
MIDI:Opened device:win32
DOS keyboard layout loaded
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
MASTER PIC ->> ISR: 00000001 | IRR: 00000000 | IMR: 11111000
SLAVE PIC ->> ISR: 00000000 | IRR: 00000000 | IMR: 11101100

For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>MOUNT C "C:\Users\MI\Desktop\BORLANDC"
Drive C is mounted as local directory C:\Users\MI\Desktop\BORLANDC\

Z:\>C:

C:\>LAB4.EXE

C:\>_
```

Рис.1 Программа на C++

## **5 Заключение**

В ходе лабораторной работы мы познакомились с работой устройства контроллера прерываний, а также изучили возможность языка С для работы с видеопамятью и изменения векторов прерываний. В результате разработки программы были реализованы следующие задачи:

- Замена стандартных векторов IRQ0-7 и IRQ8-15
- Вывод состояния регистров запросов, обслуживания и масок
- Переход к стандартным прерываниям для корректной работы системы.

Хост ОС Windows 10 x64, эмулятор DOSbox.