Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №2
«Программирование контроллера прерываний»
Вариант 23

Выполнил:
Студент группы 250501
Снитко Д.А.

Проверил:
Преподаватель
Одинец Д.Н.

Минск 2024

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

- Все векторы аппаратных прерываний ведущего и ведомого контроллера переносятся на пользовательские прерывания с помощью функций getvect и setvect.
- Производится инициализация контроллеров, заключающаяся в последовательности команд: ICW1, ICW2, ICW3 и ICW4.
- С помощью функции _dos_keep осуществляется выход в DOS, при этом программа остаётся резидентной.
- В каждом обработчике выводятся в видеопамять в двоичной форме значения регистров запросов на прерывания, обслуживаемых прерываний, масок. Затем вызываются стандартные обработчики прерываний.

3. Листинг программы

Далее приведен листинг резидентной программы, выполняющей перенос всех векторов аппаратных прерываний ведущего и ведомого контроллера на пользовательские прерывания.

```
#include <dos.h>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>

#define M_BASE_VECTOR 0x08
#define S_BASE_VECTOR 0x90

struct VIDEO
```

```
{
     unsigned char symbol;
     unsigned char attribute;
};

char color = 0xfc;

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++;
     }
     screen++;

     val = inp(0xA1);
     for (i = 0; i < 8; i++)
     {
          temp = val % 2;
          val = val >> 1;
          screen->symbol = temp + '0';
          screen++;
     }
     screen += 63;
     outp(0x20, 0x0A);

     val = inp(0x20);
     for (i = 0; i < 8; i++)
     {
          temp = val % 2;
          val = val >> 1;
          screen->symbol = temp + '0';
          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++;
     }
     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++;
        }
        screen++;

        outp(0xA0, 0x0B);
        val = inp(0xA0);
        for (i = 0; i < 8; i++)
        {
                temp = val % 2;
                val = val >> 1;
                screen->symbol = temp + '0';
                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(...) {print(); oldint9(); }
void interrupt  newint0A(...) {print(); oldint10(); }
void interrupt  newint0B(...) {print(); oldint11(); }
void interrupt  newint0C(...) {print(); oldint12(); }
void interrupt  newint0D(...) {print(); oldint13(); }
void interrupt  newint0E(...) {print(); oldint14(); }
void interrupt  newint0F(...) {print(); oldint15(); }

void interrupt  newintC0(...) {print(); oldint70(); }
void interrupt  newintC1(...) {print(); oldint71(); }
void interrupt  newintC2(...) {print(); oldint72(); }
void interrupt  newintC3(...) {print(); oldint73(); }
void interrupt  newintC4(...) {print(); oldint74(); }
void interrupt  newintC5(...) {print(); oldint75(); }
```

```
void interrupt  newintC6(...) {print(); oldint76(); }
void interrupt  newintC7(...) {print(); oldint77(); }

void initialize()
{
     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(M_BASE_VECTOR, newint08);
     setvect(M_BASE_VECTOR + 1, newint09);
     setvect(M_BASE_VECTOR + 2, newint0A);
     setvect(M_BASE_VECTOR + 3, newint0B);
     setvect(M_BASE_VECTOR + 4, newint0C);
     setvect(M_BASE_VECTOR + 5, newint0D);
     setvect(M_BASE_VECTOR + 6, newint0E);
     setvect(M_BASE_VECTOR + 7, newint0F);

     setvect(S_BASE_VECTOR, newintC0);
     setvect(S_BASE_VECTOR + 1, newintC1);
     setvect(S_BASE_VECTOR + 2, newintC2);
     setvect(S_BASE_VECTOR + 3, newintC3);
     setvect(S_BASE_VECTOR + 4, newintC4);
     setvect(S_BASE_VECTOR + 5, newintC5);
     setvect(S_BASE_VECTOR + 6, newintC6);
     setvect(S_BASE_VECTOR + 7, newintC7);

     _disable();

     outp(0x20, 0x11);
     outp(0x21, M_BASE_VECTOR);
     outp(0x21, 0x04);
     outp(0x21, 0x01);

     outp(0xA0, 0x11);
     outp(0xA1, S_BASE_VECTOR);
     outp(0xA1, 0x02);
     outp(0xA1, 0x01);

     _enable();
}
```

```
int main()
{
     unsigned far *fp;
     initialize();
     system("cls");

     printf("                    - mask\n");
     printf("                    - prepare\n");
     printf("                    - service\n");
     printf("Master   Slave\n");

     FP_SEG(fp) = _psp;
     FP_OFF(fp) = 0x2c;
     _dos_freemem(*fp);

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

     return 0;
}
```

4. Тестирование программы

Рисунок 4.1 – Результат работы программы при запуске.

5. Заключение

В ходе лабораторной удалось выполнить перенос всех векторов аппаратных прерываний ведущего и ведомого контроллера на пользовательские прерывания. Использование контроллера прерываний позволяет ускорить взаимодействие процессора с внешними устройствами.

Программа компилировалась в BorlandC и запускалась в DOS, который эмулировался с помощью Oracle VM VirtualBox.