Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №4
«Программирование часов реального времени»

Выполнил:
Студент группы 250501
Снитко Д. А.

Проверил:
Преподаватель
Одинец Д.Н.

Минск 2024

1. Постановка задачи

Первая часть (общее задание). Написать программу, которая будет считывать и устанавливать время в часах реального времени. Считанное время должно выводиться на экран в удобочитаемой форме.

Условия выполнения лабораторной работы для первой части:

1. Перед началом установки новых значений времени необходимо считывать и анализировать старший байт регистра состояния 1 на предмет доступности значений для чтения и записи. Начинать операцию записи новых значений, можно только в случае, когда этот бит установлен в '0' – то есть, регистры часов доступны.

2. После проверки доступности регистров (пункт 1), необходимо отключить внутренний цикл обновления часов реального времени, воспользовавшись старшим битом регистра состояния 2 После окончания операции установки значений этот бит должен быть сброшен, для возобновления внутреннего цикла обновления часов реального времени.

3. Новые значения для регистров часов реального времени должны вводится с клавиатуры в удобном для пользователя виде.

Вторая часть

1. Используя аппаратное прерывание часов реального времени и режим генерации периодических прерываний реализовать функцию задержки с точностью в миллисекунды (с возможностью изменения частоты генерации).

Условия выполнения данного варианта:

1.1 Задержка должна вводится с клавиатуры в миллисекундах в удобной для пользователя форме.

1.2 После окончания периода задержки программа должна сообщить об этом в любой форме. При этом зависание компьютера не допускается

1.3 Предусмотреть возможность изменения интервала генерации задержки по умолчанию (1/1024 Гц) на другое значение.

2. Используя аппаратное прерывания часов реального времени и режим будильника ЧРВ (4А использовать не желательно) реализовать функции программируемого будильника.

2.1. Время будильника вводится с клавиатуры в удобной для пользователя форме.

2.2 При срабатывания будильника программа должна сообщить об этом в любой форме. При этом зависание компьютера не допускается.

Общие условия для всей лабораторной работы.

1. В программе должно быть реализовано меню, позволяющее выбрать тестируемый функционал (установка времени, считывание времени, задержка и т.д.)

2. Весь ввод/вывод данных с/на консоль должен выполняться в удобной для пользователя форме.

3. Зависание компьютера не допускается ни в ходе работы программы, ни после ее завершения.

2. Алгоритм

Алгоритм для настройки и использования будильника на часах реального времени (RTC):

Отключить обновление часов реального времени, записав управляющее слово 0x80 в регистр B RTC.

Установить время будильника, записав значения часов, минут и секунд в регистры часов, минут и секунд RTC соответственно.

Настроить регистр B RTC для разрешения прерывания будильника, установив бит 5 в 1.

Установить новый обработчик прерывания для будильника, заменив старый обработчик прерывания 0x4A на новый.

Разрешить прерывания от контроллера прерываний, записав 0x20 в порт 0xA0.

Включить обновление часов реального времени, записав управляющее слово 0x00 в регистр B RTC.

При срабатывании будильника происходит следующее:

Срабатывает прерывание 0x4A, которое обрабатывается новым обработчиком прерывания.

В обработчике прерывания выводится сообщение об активации будильника и вызывается функция resetAlarm().

Функция resetAlarm() отключает прерывание будильника, записывая управляющее слово 0xDF в регистр B RTC.

Функция resetAlarm() сбрасывает флаг активации будильника и восстанавливает старый обработчик прерывания 0x4A.

Функция resetAlarm() завершает обработку прерывания, вызвав старый обработчик прерывания 0x4A.

Для установки интервала задержки необходимо выполнить следующие действия:

Установить новый обработчик прерывания таймера, заменив старый обработчик прерывания 0x1C на новый.

Установить значение счетчика таймера, выведя в порт 0x40 значение, полученное при делении требуемого интервала задержки (в миллисекундах) на 1193.18. Сначала вывести младший байт, а затем старший байт значения.

Разрешить прерывания от таймера, записав 0x34 в порт управляющего регистра таймера с адресом 0x43.

При срабатывании прерывания таймера происходит следующее:

Срабатывает прерывание 0x1C, которое обрабатывается новым обработчиком прерывания.

В обработчике прерывания выполняется необходимый код, например, выводится сообщение на экран.

Обработчик прерывания вызывает функцию MyDelay(), которая ждет заданный интервал времени, прежде чем вернуть управление.

Обработчик прерывания завершает обработку прерывания, вызвав старый обработчик прерывания 0x1C.

Для сброса интервала задержки необходимо выполнить следующие действия:

Отключить прерывания от таймера, записав 0x36 в порт управляющего регистра таймера с адресом 0x43.

Восстановить старый обработчик прерывания таймера, вызвав функцию setvect().

Сбросить значение счетчика таймера, выведя в порт 0x40 значение 0x00.

3. Листинг программы

Далее приведен листинг программы, реализующей все поставленные задачи.

```
#include <io.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

unsigned char data[7]; // данные часов
unsigned int delayTime = 0;
unsigned int delayInterval = 1; // интервал генерации задержки в миллисекундах
unsigned int registerArray[] = { 0x00, 0x02, 0x04, 0x06, 0x07, 0x08, 0x09 };
int alarmOn = 0;
char* months[] =
{
    "JANUARY",
    "FEBRUARY",
    "MARCH",
    "APRIL",
    "MAY",
    "JUNE",
    "JULY",
    "AUGUST",
    "SEPTEMBER",
    "OCTOBER",
    "NOVEMBER",
    "DECEMBER"
};

void interrupt newTime(...);  // новый обработчик прерываний часов
void interrupt newAlarm(...); // новый обработчик прерываний будильника
void interrupt(*lastTime)(...); // старое прерывание часов
void interrupt(*lastAlarm) (...); // старое прерывание будильника
```

```
void Menu();
void ShowTime();
int ConvertToDecimal(int BCD);
int convertToBCD(int decimal);
void setTime();
void MyDelay(unsigned int delayMs);
void enterTime();
void setAlarm();
void resetAlarm();
void setDelayInterval(unsigned int interval);

int main()
{
    Menu();
    return 0;
}

void Menu() {
    while (1) {
        system("cls");
        ShowTime();
        printf("\n1 - Set time  2 - Set delay  3 - Set delay interval  4 - Set alarm
0 - Exit");
        if(alarmOn == 1) printf("\n\nALARM ON");
        if(alarmOn == 2) {
            printf("\n\nALARM! ALARM! ALARM!");
            delay(5000);
            alarmOn = 0;
        }

        printf("\n\nEnter choice: ");
        delay(1000);
        if (kbhit()) {
            switch(getch()) {
            case '0':
                return;

            default:
                break;

            case '1':
                system("cls");
                setTime();
                break;

            case '2':
                system("cls");
                unsigned int delayMs = 0;
                printf("Input delay (ms): ");
                scanf("%d", &delayMs);
                MyDelay(delayMs);
                break;

            case '3':
                system("cls");
```

```
                unsigned int interval = 0;
                printf("Input delay interval (ms): ");
                scanf("%d", &interval);
                setDelayInterval(interval);
                break;

            case '4':
                system("cls");
                setAlarm();
                break;
            }
        }
    }
}

void ShowTime() {

    int i = 0;
    for (i = 0; i < 7; i++) {
        outp(0x70, registerArray[i]); // выбор адреса в памяти CMOS
        data[i] = inp(0x71); // считывание значения по адресу в массив
    }

    int decimalData[7]; // перевод значений в десятичный вид
    for (i = 0; i < 7; i++) {
        decimalData[i] = ConvertToDecimal(data[i]);
    }

    // вывод на экран
    printf("%02d:%02d:%02d %02d.%02d.%02d\n", decimalData[2], decimalData[1], deci-
malData[0], decimalData[3], decimalData[4], decimalData[5]);
}

int ConvertToDecimal(int BCD) {
    return ((BCD / 16 * 10) + (BCD % 16));
}

int ConvertToBCD(int decimal)
{
    return ((decimal / 10 * 16) + (decimal % 10));
}

void setTime()
{
    enterTime(); // ввод нового времени
    disable(); // запрет на прерывание

    // проверка на доступность значений для чтения/записи
    unsigned int check;
    do {
        outp(0x70, 0xA); // выбор регистра A
        check = inp(0x71) & 0x80; // 0x80 - 1000 0000
        // 7-й бит в 1 для обновления времени
    } while (check);
```

```
    // отключение обновления часов реального времени
    outp(0x70, 0xB); // выбор регистра B
    outp(0x71, inp(0x71) | 0x80); // 0x80 - 1000 0000
    // 7-й бит в 1 для запрета обновления часов

    for (int i = 0; i < 7; i++) {
        outp(0x70, registerArray[i]); // выбор нужного значения данных
        outp(0x71, data[i]); // подача в регистр нужного значения
    }

    // включение обновления часов реального времени
    outp(0x70, 0xB); // выбор регистра B
    outp(0x71, inp(0x71) & 0x7F); // 0x7F - 0111 1111
    // 7-й бит в 0 для разрешения обновления часов

    enable(); // разрешение на прерывание
    system("cls");
}

void enterTime() {
    int enter;

    do {
        rewind(stdin);
        printf("Enter century: ");
        scanf("%i", &enter);
    } while ((enter > 99 || enter < 0));
    data[6] = ConvertToBCD(enter);

    do {
        rewind(stdin);
        printf("Enter year: ");
        scanf("%i", &enter);
    } while ((enter > 99 || enter < 0));
    data[5] = ConvertToBCD(enter);

    do {
        rewind(stdin);
        printf("Enter month: ");
        scanf("%i", &enter);
    } while ((enter > 12 || enter < 1));
    data[4] = ConvertToDecimal(enter);

    do {
        rewind(stdin);
        printf("Enter day: ");
        scanf("%i", &enter);
    } while ((enter > 31 || enter < 1));
    data[3] = ConvertToBCD(enter);

    do {
        rewind(stdin);
        printf("Enter hours: ");
        scanf("%i", &enter);
    } while ((enter > 23 || enter < 0));
```

```
    data[2] = ConvertToBCD(enter);

    do {
        rewind(stdin);
        printf("Enter minutes: ");
        scanf("%i", &enter);
    } while (enter > 59 || enter < 0);
    data[1] = ConvertToBCD(enter);

    do {
        rewind(stdin);
        printf("Enter seconds: ");
        scanf("%i", &enter);
    } while (enter > 59 || enter < 0);
    data[0] = ConvertToBCD(enter);
}

void MyDelay(unsigned int delayMs)
{
    disable(); // запрет на прерывание

    // установка нового обработчика прерываний
    lastTime = getvect(0x70);
    setvect(0x70, newTime);

    enable(); // разрешение на прерывание

    // размаскировка линии сигнала запроса от ЧРВ
    // 0xA1 - новое значение счетчика для системного таймера
    outp(0xA1, inp(0xA1) & 0xFE); // 0xFE = 1111 1110
    // 0-й бит в 0 для разрешения прерывания от ЧРВ

    outp(0x70, 0xB); // выбор регистра B
    outp(0x71, inp(0x71) | 0x40); // 0x40 = 0100 0000
    // 6-й бит регистра B установлен в 1 для периодического прерывания

    delayTime = 0;
    while (delayTime <= delayMs);
    setvect(0x70, lastTime);
    return;
}

void setAlarm()
{
    enterTime(); // ввод нового времени
    disable(); // запрет на прерывание

    // проверка на доступность значений для чтения/записи
    unsigned int check;
    do {
        outp(0x70, 0xA); // выбор регистра A
        check = inp(0x71) & 0x80; // 0x80 - 1000 0000
        // 7-й бит в 1 для обновления времени
    } while (check);
```

```
    // установка часов в регистр будильника
    outp(0x70, 0x05);
    outp(0x71, data[2]);

    // установка минут в регистр будильника
    outp(0x70, 0x03);
    outp(0x71, data[1]);

    // установка секунд в регистр будильника
    outp(0x70, 0x01);
    outp(0x71, data[0]);

    // установка бита будильника в регистре B
    outp(0x70, 0xB); // выбор регистра B
    outp(0x71, (inp(0x71) | 0x20)); // 0x20 - 0010 0000
    // 5-й бит регистра B установлен в 1 для разрешения прерывания будильника

    // переопределение прерывания будильника
    lastAlarm = getvect(0x70); // 0x70 - IRQ8 (CMOS Real-Time Clock)
    setvect(0x70, newAlarm);

    enable(); // разрешение на прерывание
    alarmOn = 1;
}

void resetAlarm()
{
    // проверка на наличие установленного будильника
    if (lastAlarm == NULL)
        return;

    disable(); // запрет на прерывание

    // возврат старого прерывания
    setvect(0x70, lastAlarm);  // 0x70 - IRQ8 (CMOS Real-Time Clock)

    // проверка на доступность значений для чтения/записи
    unsigned int check;
    do {
        outp(0x70, 0xA); // выбор регистра A
        check = inp(0x71) & 0x80;  // 0x80 - 1000 0000
        // 7-й бит в 1 для обновления времени
    } while (check);

    // запись нулевых значений в регистр будильника
    outp(0x70, 0x05); // 0x05 - часы
    outp(0x71, 0x00);

    outp(0x70, 0x03); // 0x03 - минуты
    outp(0x71, 0x00);

    outp(0x70, 0x01); // 0x01 - секунды
    outp(0x71, 0x00);

    // сброс бита будильника в регистре B
```

```
    outp(0x70, 0xB); // выбор регистра B
    outp(0x71, (inp(0x71) & 0xDF)); // 0xDF - 1101 1111
    // 5-й бит в 0 для запрета прерывания будильника

    enable(); // разрешение на прерывание
}

void interrupt newTime(...) // новый обработчик прерываний часов
{
    delayTime++;
    outp(0x70, 0x0C); // выбор адреса в памяти CMOS (запись)
    inp(0x71); // данные по этому адресу (запись/чтение)
    // посыл сигнала контроллерам прерываний об окончании прерывания
    outp(0x20, 0x20);
    outp(0xA0, 0x20);
}

void interrupt newAlarm(...) // новый обработчик прерываний будильника (IRQ8)
{
    system("cls");
    alarmOn = 2;
    lastAlarm();
    resetAlarm();
}

void setDelayInterval(unsigned int interval)
{
    delayInterval = interval;
}
```

4. Тестирование программы

Во время работы программы Пользователь может видеть время, установить его, изменить задержку, изменить интервал задержки, установить будильник.

```
12:31:54 05.25.04

1 - Set time  2 - Set delay  3 - Set delay interval  4 - Set alarm  0 - Exit

Enter choice: _
```

Рисунок 4.1 – Результат работы программы

5. Заключение

В результате выполнения данной работы был разработан программный модуль для управления реальными часами (RTC) и будильником на языке Си с использованием прерываний. Были реализованы функции для установки времени, задержки, интервала задержки, а также установки и сброса будильника. Программа компилировалась в BorlandC и запускалась в DOS, который эмулировался с помощью DosBox.