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

Кафедра электронных вычислительных машин

Лабораторная работа № 5
«Программирование часов реального времени»

Проверил:
Одинец Д. Н.

Выполнил:
ст. гр. 950503
Полховский А.Ф.

Минск 2021

1. Постановка задачи

1. Написать программу, которая будет считывать и устанавливать время в часах реального времени. Считанное время должно выводиться на экран в удобочитаемой форме.
2. Используя аппаратное прерывание часов реального времени и режим генерации периодических прерываний реализовать функцию задержки с точностью в миллисекунды.

2. Алгоритм решения задачи

Для того чтобы реализовать поставленные задачи необходимо:

Схема работы с CMOS:

1. Заносим в 0x70 адреса интересующих нас участков cmos памяти.
2. Считываем / записываем 0x71 как значение адреса из cmos.

Установить время:

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. Считываем по адресу 0x0, 0x2, 0x4 секунды, минуты, часы и выводим на экран
3. После установки значений времени нужно возобновить внутренний цикл обновления часов реального времени.

Задержка:

Для реализации функции задержки заменён обработчик прерывания 0x70, в котором происходит отсчёт миллисекунд. Для включения периодического прерывания, происходящего примерно миллисекунду, 6-й бит регистра B устанавливается в '1':

1. Отмаскируем периодическое прерывание.
2. Ставим вектор прерывания, который инкрементирует наше значение счетчика в миллисекундах.
3. Ожидаем пока счетчик меньше нашей заданной задержки.
4. Возвращаем старый обработчик.

3. Листинг программы

```
#include <dos.h>
#include <iostream.h>
#include <stdlib.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include <wctype.h>
```

```
//using namespace std;

unsigned long millisecondsTimer = 0;

int hour;
int minute;
int second;

int value;
int input;

void interrupt(*originalInterruption)(...);
void interrupt newInterruption(...);

int intToBCD(int);
int BCDToInt(int);
void showTime();
void setTime();
void setDelay();

int main() {
    clrscr();

    while (1) {
        cout    << endl
                << "1. Get time"        << endl
                << "2. Set time"        << endl
                << "3. Delay"           << endl
                << "0. Exit program"    << endl;

        cin >> input;

        switch (input){
            case 1:
                showTime();
                break;
            case 2:
                setTime();
                break;
            case 3:
                setDelay();
                break;
            case 0:
                return 0;
        }
    }
    return 0;
}

void interrupt newInterruption(...){
    millisecondsTimer++;
```

```
    outp(0x70, 0x0C);
    inp(0x71);

    outp(0x20, 0x20); //send EOI to Master interruption controller
    outp(0xA0, 0x20); //send EOI to Slave interruption controller
}

int BCDToInt(int number) {
    number = number;
    return (((number / 0x10) * 10) + (number % 0x10));
}

int intToBCD(int number) {
    number = number;
    return (((number / 10) * 0x10) + (number % 10));
}

void showTime(){
    //seconds
    do {
        outp(0x70, 0x0A);
    } while (inp(0x71) & 0x80);     // 0x80 == 10000000, seventh bit == 1 -> clock is busy

    outp(0x70, 0);
    second = inp(0x71);

    //minute
    do {
        outp(0x70, 0x0A);
    } while (inp(0x71) & 0x80);

    outp(0x70, 2);
    minute = inp(0x71);

    //hours
    do {
        outp(0x70, 0x0A);
    } while (inp(0x71) & 0x80);

    outp(0x70, 4);
    hour = inp(0x71);

    cout << BCDToInt(hour) << ':' << BCDToInt(minute) << ':' << BCDToInt(second);
}

void setTime() {
    do {
        outp(0x70, 0x0A);
    } while (inp(0x71) & 0x80);     //while busy

    outp(0x70, 0x0B);
```

```
    value = inp(0x71) | 0x80;       //set seventh bit in B status byte to '1' to disable clock
updating
    outp(0x70, 0x0B);
    outp(0x71, value);              //write chenged value

    cout << "Enter time:" << endl;
    cout << "Hours:";
    cin >> hour;
    cout << "Minute:";
    cin >> minute;
    cout << "Second:";
    cin >> second;

    outp(0x70, 0x04);
    outp(0x71, intToBCD(hour));
    outp(0x70, 0x02);
    outp(0x71, intToBCD(minute));
    outp(0x70, 0x00);
    outp(0x71, intToBCD(second));

    //enable clock updating
    do {
        outp(0x70, 0x0A);
    } while (inp(0x71) & 0x80);

    outp(0x70, 0x0B);
    value = inp(0x71) & 0x7F;
    outp(0x71, value);
}

void setDelay() {
    unsigned long millisecondsDelay = 0;

    //change interruption
    disable();  //cli
    originalInterruption = getvect(0x70);
    setvect(0x70, newInterruption);
    enable();   //sti

    cout << "Enter delay in milliseconds: ";
    cin >> millisecondsDelay;

    cout << "Start time:";
    showTime();



    value = inp(0xA1);
    outp(0xA1, value & 0xFE);       //0xFE == 11111110, zero bit == 0 to enable RTC
interruptions

    //enabling periodic interrupts
```

```
    outp(0x70, 0x0B);              //second status byte
    value = inp(0x71);
    outp(0x70, 0x0B);
    outp(0x71, value | 0x40);      //0x40 == 01000000, sixth bit == 1 -> enable periodic
interruptions IRQ-8

    millisecondsTimer = 0;
    while (millisecondsTimer != millisecondsDelay) {};
    cout << millisecondsDelay <<  "milliseconds passed!" << endl;
    setvect(0x70, originalInterruption);    //return original interruption

    outp(0x70, 0x0B);
    value = inp(0x71) & 0x7F;      //0x7F == 01111111, seventh bit == 1 ->
    outp(0x71, value);

    cout << "End time:";
    showTime();
}
```

4. Заключение

CMOS память является удобным инструментом управлением времени, который позволяет использовать будильник, а также устанавливать и считывать время. С помощью этого можно конструировать программы, упрощающие тайм-менеджмент.

В программе реализовано меню, позволяющее выбрать тестируемый функционал (установка времени, считывание времени, задержка): '1' – вывод текущего времени, '2' – установка времени, '3' – задержка в миллисекундах. Выход из программы производится по нажатию Esc.