Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №5
«Программирование часов реального времени»
Вариант 5

Выполнил:
Студент группы 950501
Деркач А.В.

Проверил:
Преподаватель
Одинец Д.Н.

Минск, 2021

1. Постановка задачи

Написать программу, которая будет считывать и устанавливать время в часах реального времени. Считанное время должно выводиться на экран в удобочитаемой форме.

1. Используя аппаратное прерывание часов реального времени и режим генерации периодических прерываний реализовать функцию задержки с точностью в миллисекунды.
2. Используя аппаратное прерывания часов реального времени и режим будильника реализовать функции программируемого будильника.

2. Алгоритм

Перед установкой значений времени вызывается функция, которая считывает и анализирует старший байт регистра состояния 1 на предмет доступности значений для чтения и записи. Когда этот бит установлен в '0', отключается внутренний цикл обновления часов реального времени: для этого старший бит регистра состояния 2 устанавливается в '1'.

Считывание или запись значений времени происходит следующим образом: в порт 70h отправляется индекс регистра CMOS, соответствующий значению времени (секунды, часы и т. д.), затем происходит чтение значения из порта 71h (или запись значения в порт).

После установки значений времени вызывается функция, которая возобновляет внутренний цикл обновления часов реального времени.

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

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>

char data[6]; // данные часов
unsigned int delayTime = 0;
unsigned int registerArray[] = { 0x00, 0x02, 0x04, 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();

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

void Menu() {
     while (1) {
          system("cls");
          ShowTime();
          printf("\n1 - Set time");
          printf("\n2 - Set delay");
          printf("\n3 - Set alarm");
          printf("\n0 - 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");
                    int delayMs = 0;
                    printf("Input delay (ms): ");
                    scanf("%d", &delayMs);
                    MyDelay(delayMs);
                break;

                case '3':
                    system("cls");
                    setAlarm();
                break;
            }
            }
      }
}

void ShowTime() {

    int i = 0;
    for (i = 0; i < 6; i++) {
        outp(0x70, registerArray[i]); // выбор адреса в памяти CMOS
        data[i] = inp(0x71); // считывание значения по адресу в массив
    }

    int decimalData[6]; // перевод значений в десятичный вид
    for (i = 0; i < 6; i++) {
        decimalData[i] = ConvertToDecimal(data[i]);
    }

    // вывод на экран
      if (decimalData[2] < 10) printf("0%1d", decimalData[2]); //часы
            else printf("%2d", decimalData[2]);
      if (decimalData[1] < 10) printf(":0%1d", decimalData[1]); //минуты
            else printf(":%2d", decimalData[1]);
      if (decimalData[0] < 10) printf(":0%1d", decimalData[0]); //секунды
            else printf(":%2d", decimalData[0]);
    printf("\n%2d %s 20%2d\n", decimalData[3], months[decimalData[4] - 1],
decimalData[5]); // день, месяц, год
}

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 < 6; 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 year: ");
        scanf("%i", &enter);
    } while ((enter > 100 || enter < 21));
    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 minuts: ");
        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]);

    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(0x4A); // 0x4A - 1001 010 (обновление времени)
    setvect(0x4A, newAlarm); // 0x4A - текущая дата и время в формате BCD
    outp(0xA1, (inp(0xA0) & 0xFE)); // 0xFE - 1111 1110
    // 0-й бит в 0 для разрешения прерывания от ЧРВ

    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(0x4A, lastAlarm);  // 0x4A - текущая дата и время в формате BCD
    outp(0xA1, (inp(0xA0) | 0x01)); // 0x01 - 0000 0001 (пересчет частоты
прерывания)

    // 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);

    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
{
     system("cls");
     alarmOn = 2;
    lastAlarm();
    resetAlarm();
}
```

4. Тестирование программы

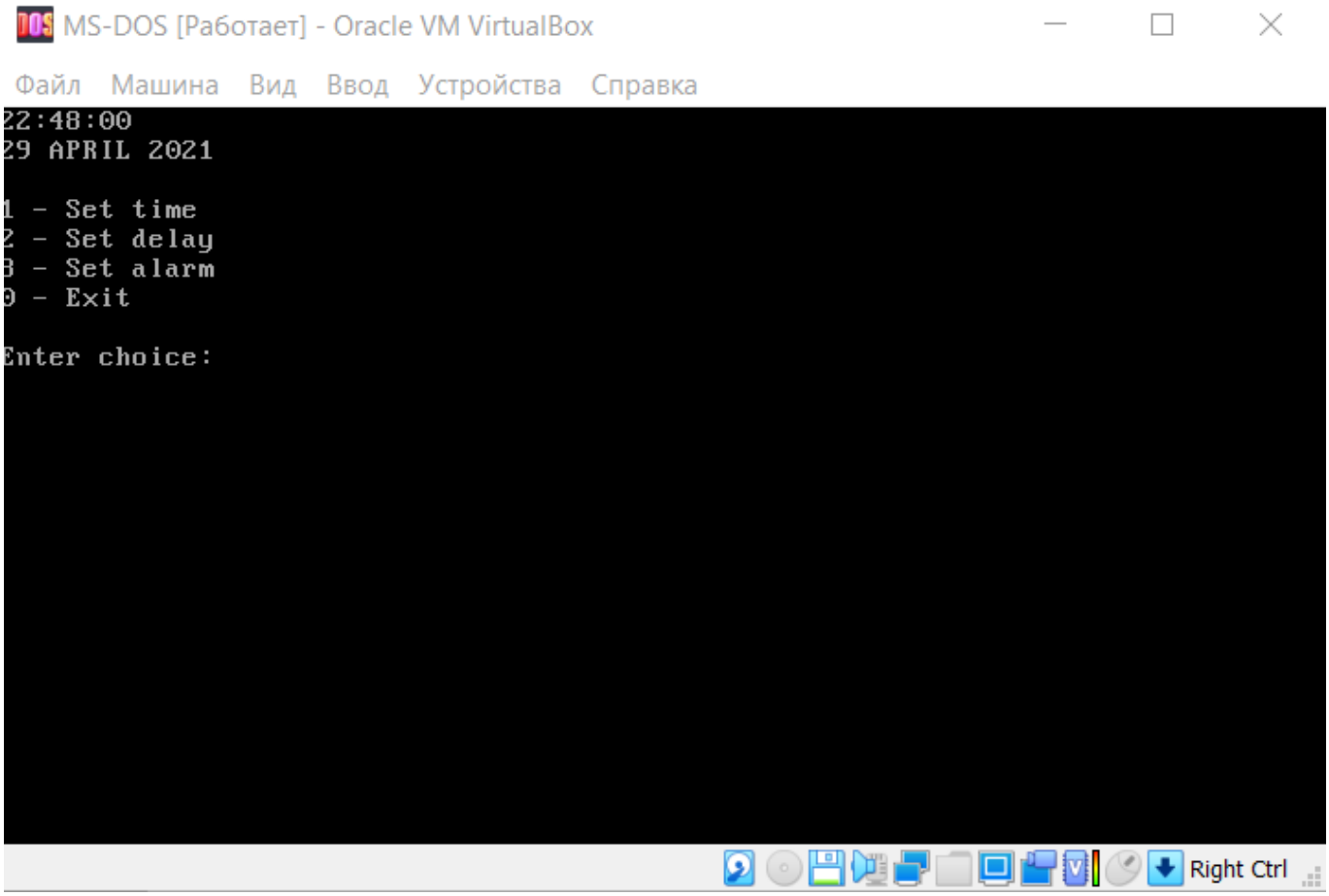

Рисунок 4.1 — Меню пользователя с выводом текущего времени.

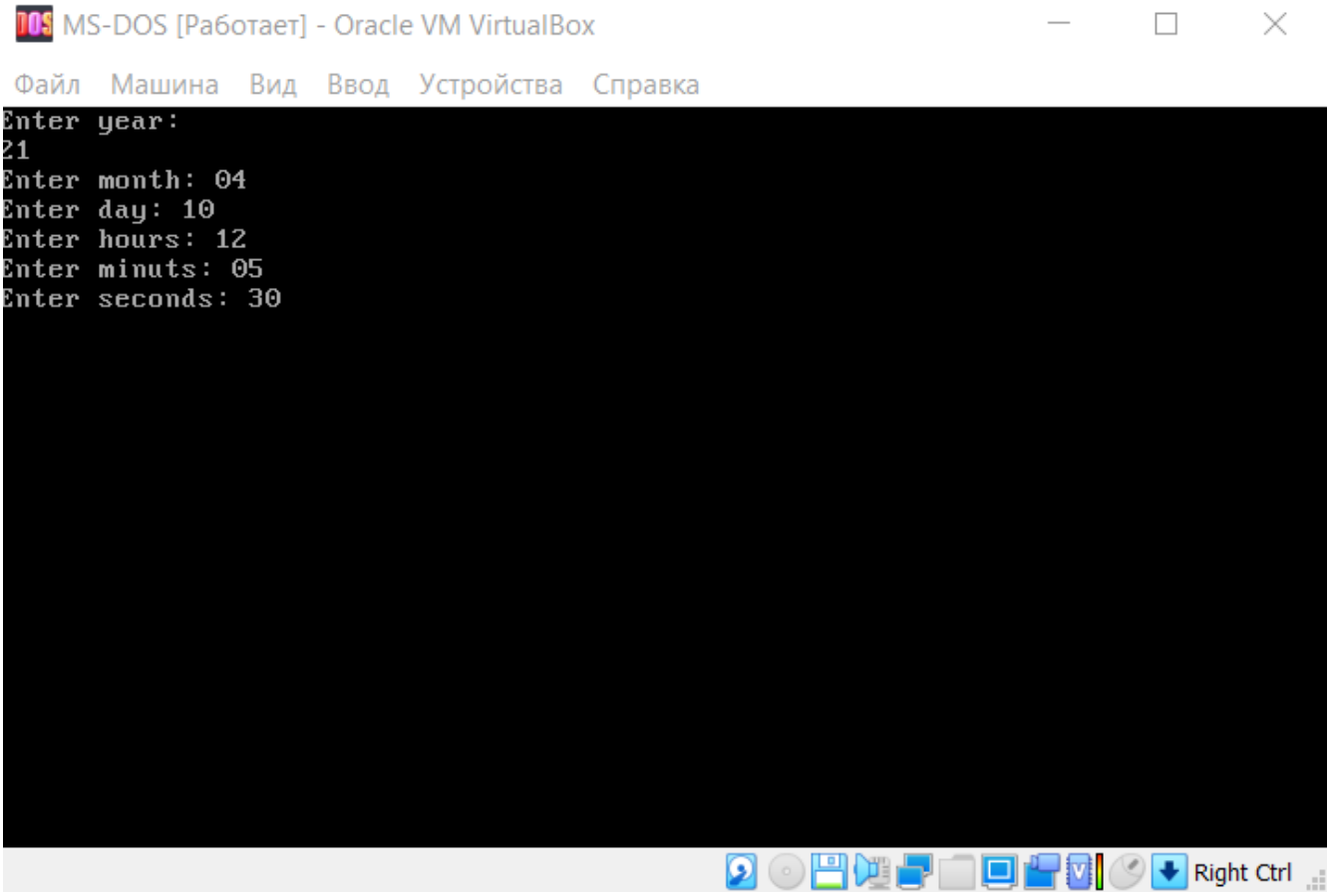

Рисунок 4.2 — Установка нового времени

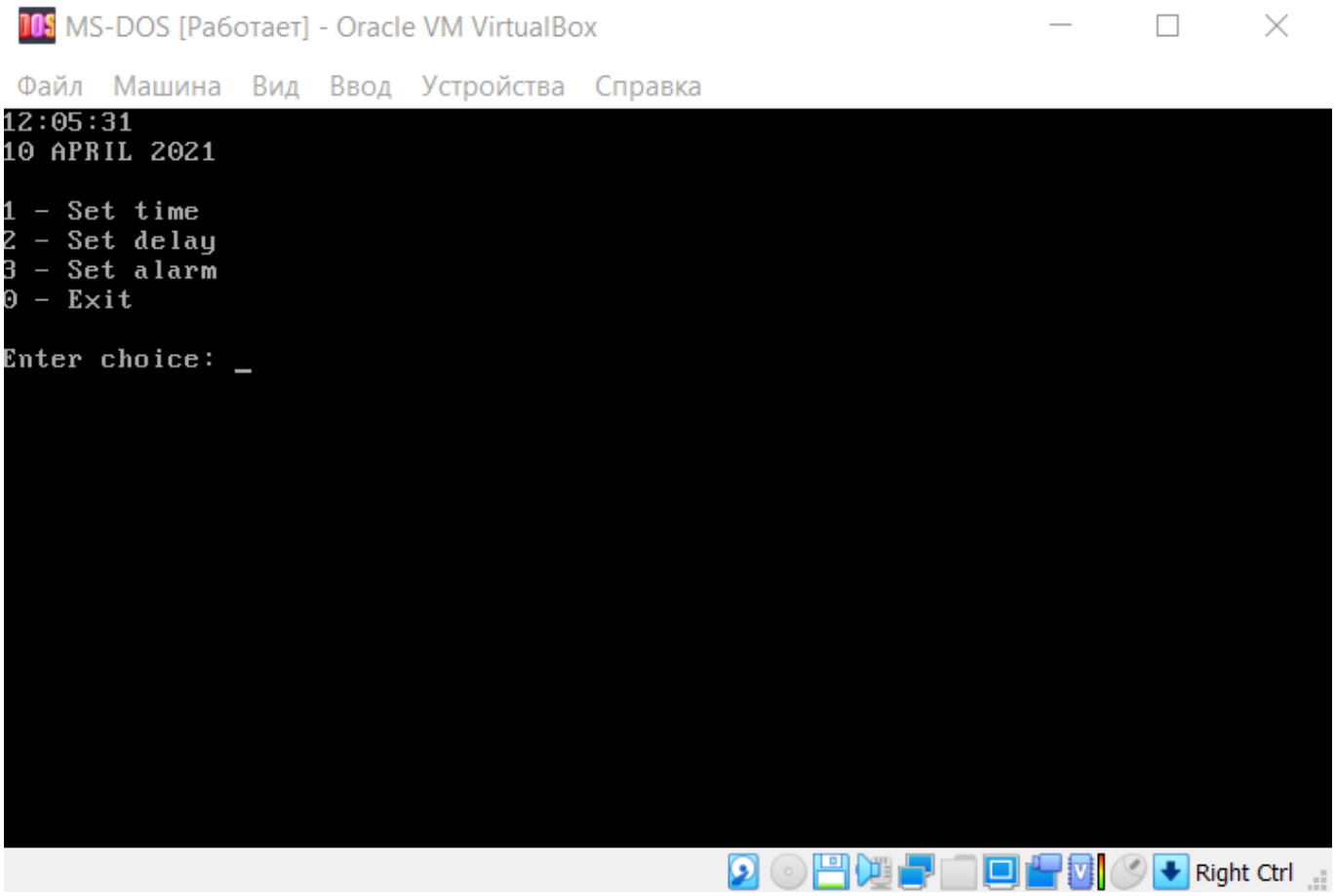

Рисунок 4.3 — Вывод нового времени.

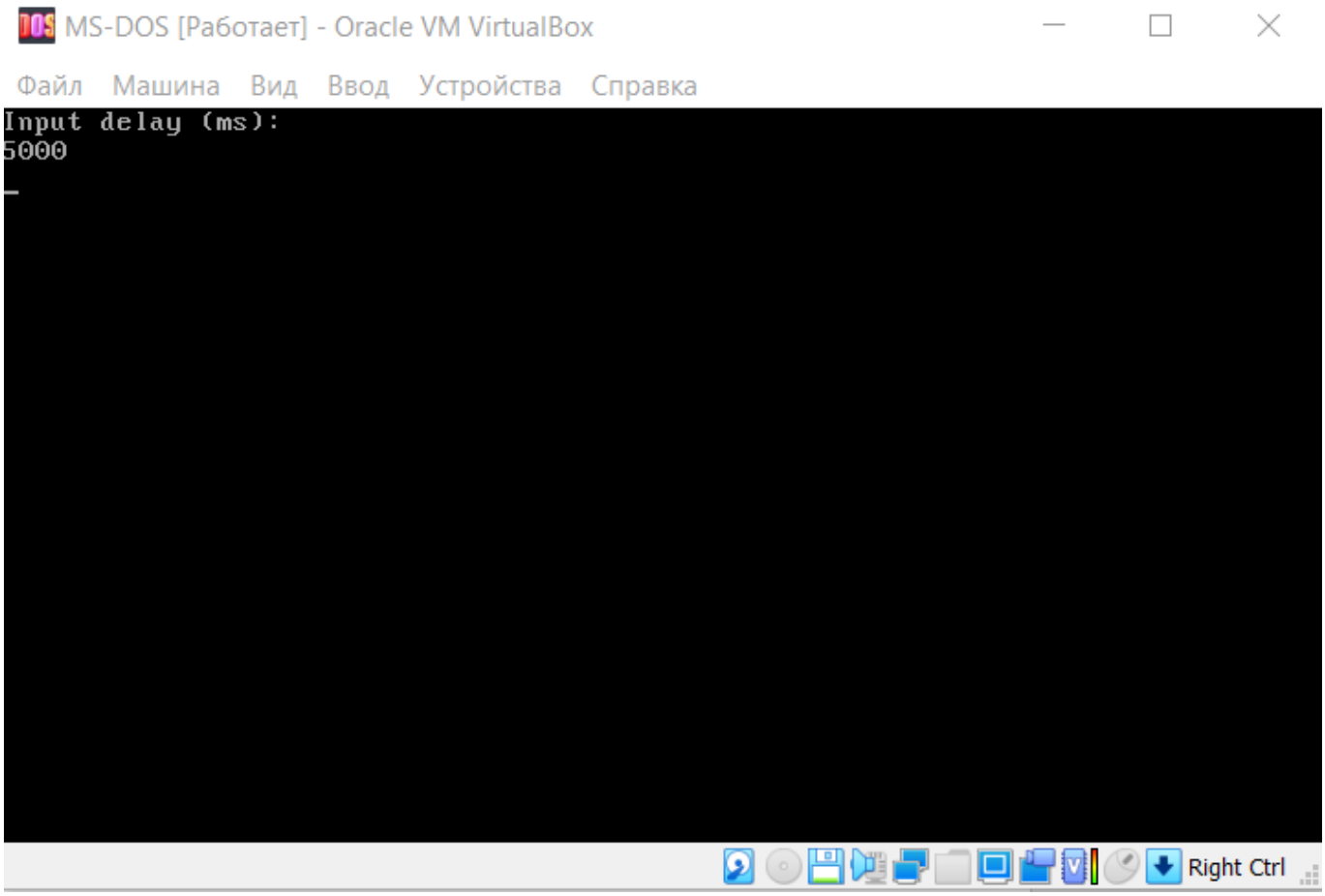

Рисунок 4.4 — Установка задержки.

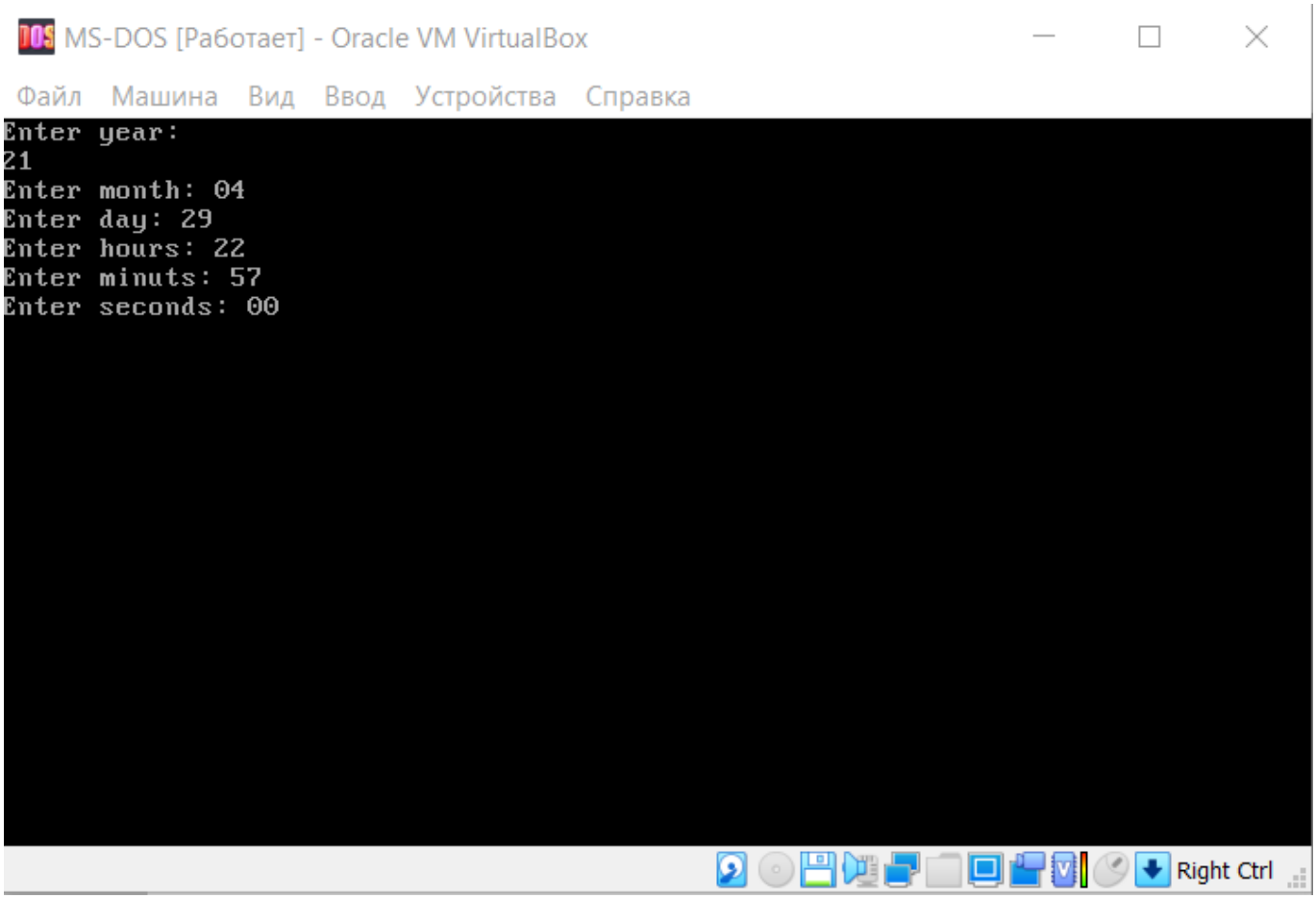

Рисунок 4.5 — Установка будильника.

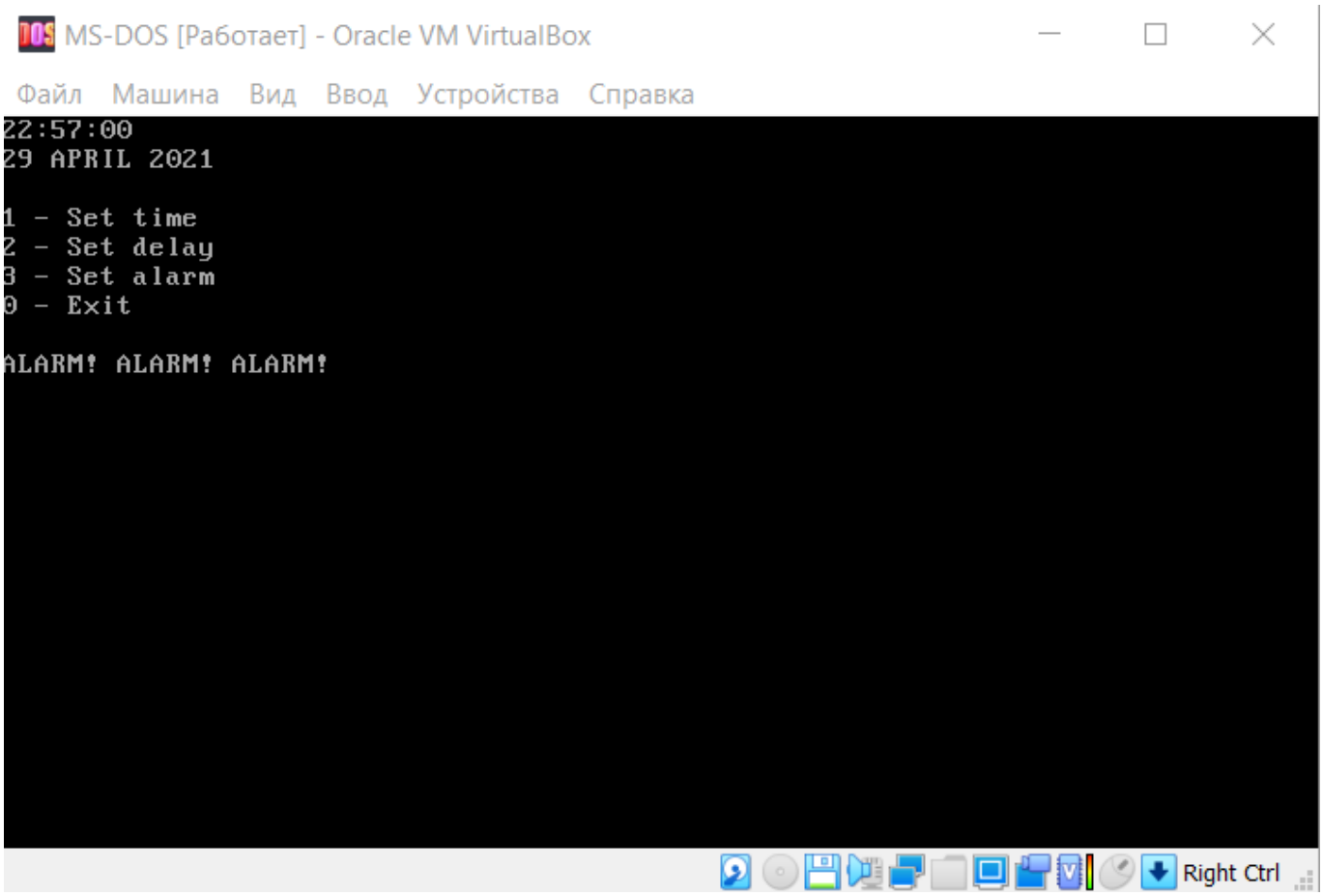

Рисунок 4.6 — Срабатывание будильника.

5. Заключение

В данной лабораторной работе были выполнены все поставленные задачи: написана программа, которая считывает и устанавливает время в часах реального времени, реализована функция задержки, используя аппаратное прерывание часов реального времени и режим генерации периодических прерываний, а также была реализована функция программируемого будильника, используя аппаратное прерывания часов реального времени и режим будильника.

Программа компилировалась в Turbo C++ и запускалась в DOS, который эмулировался с помощью VirtualBox.