Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №4
«Программирование часов реального времени»
Вариант 10

Выполнил:
Студент группы 150501
Кипятков В. И.

Проверил:
Преподаватель
Одинец Д.Н.

Минск, 2023

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


unsigned int delayTime = 0;
unsigned int delayMs;

unsigned int date[6];
unsigned int dateReg[] = { 0x00, 0x02, 0x04, 0x07, 0x08, 0x09 };


void interrupt(*oldDelay)(...);
void interrupt(*oldAlarm) (...);
```

```
void getTime();
void setTime();
void delay(unsigned int);
void setAlarm();
void inputTime();
unsigned int bcdToDec(unsigned int);
unsigned int decToBcd(unsigned int);
void pByte(unsigned int byte);
void changeFreqency();

void interrupt newDelay(...)
{
    delayTime++;
    outp(0x70, 0x0C);
    inp(0x71);

    outp(0x20, 0x20);//00101000
    outp(0xA0, 0x20);
    if (delayTime == delayMs)
    {
        puts("Delay's end");
        disable();
        setvect(0x70, oldDelay);
        enable();
        outp(0x70, 0x0B);
        outp(0x71, inp(0x71) & 0xBF);// 1011 1111
    }
}


void interrupt newAlarm(...)
{
    puts("Alarm!!!");

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

    outp(0x20, 0x20);
    outp(0xA0, 0x20);

    disable();
    setvect(0x70, oldAlarm);
    enable();
    outp(0x70, 0x0B);
    outp(0x71, inp(0x71) & 0xDF); // 1101 1111
}

int main()
{
    while (1) {
        printf("1 - Time\n");
        printf("2 - Set time\n");
        printf("3 - Set alarm\n");
        printf("4 - Set delay\n");
        printf("5 - Set freq\n");
        printf("0 - Exit\n\n");

        switch (getch()) {
        case '1':
            getTime();
            break;

        case '2':
            setTime();
            break;

        case '3':
            setAlarm();
```

```
            break;

        case '4':
            fflush(stdin);
            printf("Input delay (ms): ");
            scanf("%u", &delayMs);
            delay(delayMs);
            break;
        case '5':
            changeFreqency();
            break;
        case '0':
            printf("\n\n");
            return 0;
        default:
            printf("\n\n");
            break;
        }
    }
}

void pByte(unsigned int byte)
{
    unsigned char bit;
    for (int i = 7; i >= 0; i--)
    {
        bit = byte & 1;
        byte /= 2;
        printf("%c", bit + '0');
    }
    printf("\n");
}
void getTime()
{
   unsigned char state;
    printf("\n\n");
    int i = 0;
    for (i = 0; i < 6; i++)
    {
        outp(0x70, 0x0A);
        state = inp(0x71);
        if (state >> 7)
        {
            i--;
            continue;
        }
        outp(0x70, dateReg[i]);
        date[i] = inp(0x71);
        date[i] = bcdToDec(date[i]);
    }

    printf("Date:\n%02u.%02u.%02u", date[2], date[1], date[0]);
    printf(" %02u.%02u.20%02u\n\n",  date[3], date[4], date[5]);
}
void setTime()
{
    inputTime();

    disable();

    do { outp(0x70, 0x0A); } while (inp(0x71) >> 7);

    outp(0x70, 0x0B);
    outp(0x71, inp(0x71) | 0x80); // 1000 0000

    for (int i = 0; i < 3; i++)
    {
        outp(0x70, dateReg[i]);
```

```
        outp(0x71, date[i]);
    }

    outp(0x70, 0x0B);
    outp(0x71, inp(0x71) & 0x7F); // 0111 1111

    enable();
    printf("\n\n");

}
void inputTime()
{
    int n;

    do {
        fflush(stdin);
        printf("Hours: ");
    } while ((scanf("%d", &n) != 1 || n > 23 || n < 0));
    date[2] = decToBcd(n);

    do {
        fflush(stdin);
        printf("Minutes: ");
    } while (scanf("%d", &n) != 1 || n > 59 || n < 0);
    date[1] = decToBcd(n);

    do {
        fflush(stdin);
        printf("Seconds: ");
    } while (scanf("%d", &n) != 1 || n > 59 || n < 0);
    date[0] = decToBcd(n);
}

void delay(unsigned int ms)
{
    delayTime = 0;

    disable();

    oldDelay = getvect(0x70);
    setvect(0x70, newDelay);
    outp(0xA1, (inp(0xA0) & 0xFE));
    enable();

    outp(0x70, 0x0B);
    outp(0x71, inp(0x71) | 0x40);  // 0x40 -- 01000000

    return;
}


void setAlarm()
{
    unsigned int alarmReg[] = { 0x01, 0x03, 0x05 };

    inputTime();

    disable();

    oldAlarm = getvect(0x70);
    setvect(0x70, newAlarm);
    outp(0xA1, (inp(0xA0) & 0xFE));

    do
    {
        outp(0x70, 0x0A);
    } while (inp(0x71) >> 7);
```

```
    for (int i = 0; i < 3; i++)
    {
        outp(0x70, alarmReg[i]);
        outp(0x71, date[i]);
    }

    enable();

    outp(0x70, 0x0B);
    outp(0x71, inp(0x71) | 0x20); //00100000

    printf("Alarm enable\n\n");
}

void changeFreqency()
{
    int freq;
    int q;
    int bin;
    puts("set freq:");
    puts("1 - 2 hertz");
    puts("2 - 4 hertz");
    puts("3 - 8 hertz");
    puts("4 - 16 hertz");
    puts("5 - 32 hertz");
    puts("6 - 64 hertz");
    puts("7 - 128 hertz");
    puts("8 - 256 hertz");
    puts("9 - 512 hertz");
    puts("10 - 1024 hertz");
    puts("11 - 2048 hertz");
    puts("12 - 4096 hertz");
    puts("13 - 8192 hertz");
    scanf("%d", &q);
    freq = 0x0F - q + 1;

    outp(0x70, 0x0A);
    bin = inp(0x71);
    printf("before =%X= \n", bin);

    outp(0x70, 0x0A);
    outp(0x71, inp(0x71) | 0x80); // 1000 0000

    outp(0x70, 0x0A);
    outp(0x71, inp(0x71) & (0x20 + freq));

    outp(0x70, 0x0A);
    outp(0x71, inp(0x71) & 0x7f);

    outp(0x70, 0x0A);
    bin = inp(0x71);  //00100110
    printf("\nafter =%X= \n", bin);
}

unsigned int bcdToDec(unsigned int bcd)
{
    return (bcd / 16 * 10) + (bcd % 16);
}
unsigned int decToBcd(unsigned int dec)
{
    return (dec / 10 * 16) + (dec % 10);
}
```

4. Тестирование программы

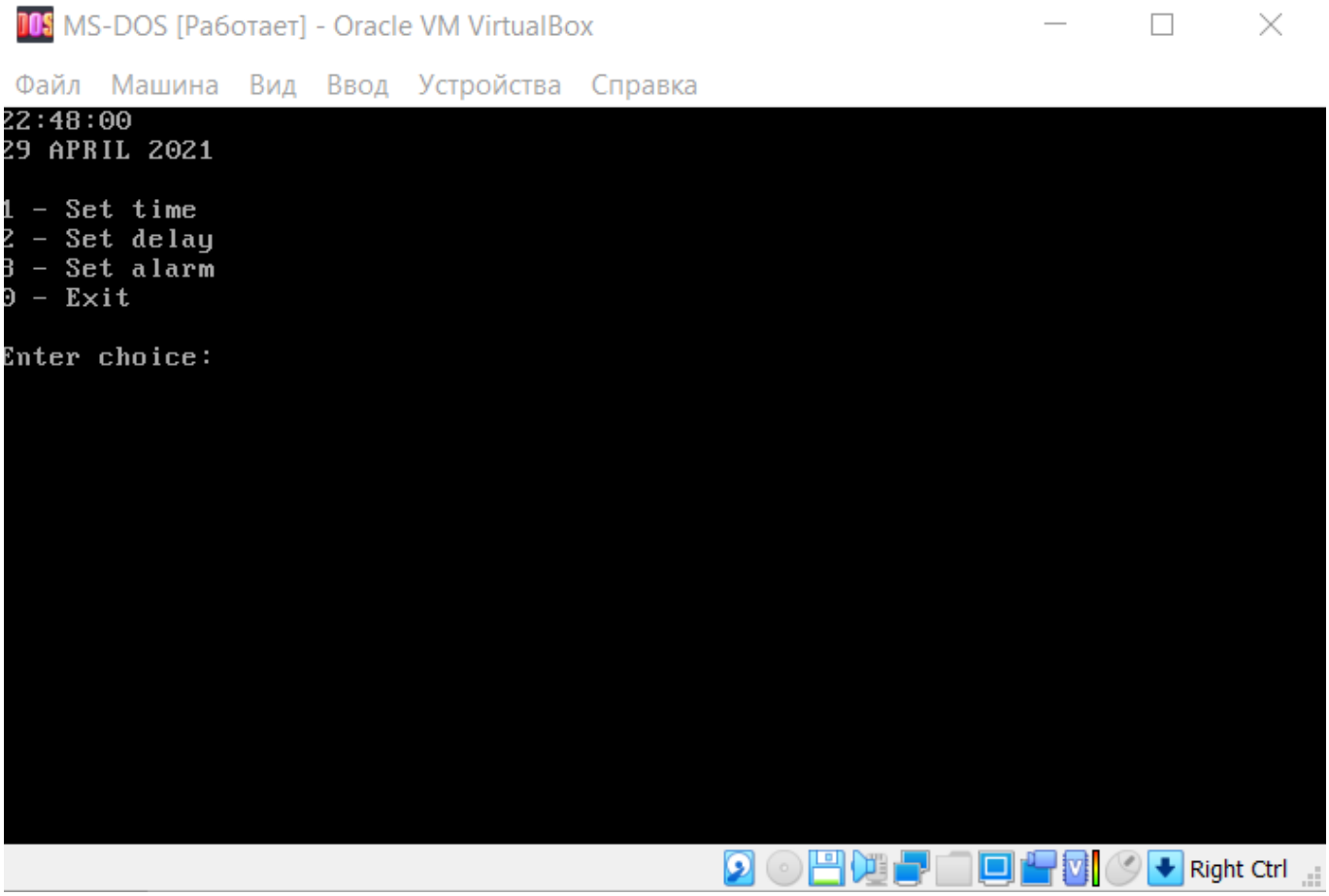

Рисунок 4.1 — Меню пользователя с выводом текущего времени.

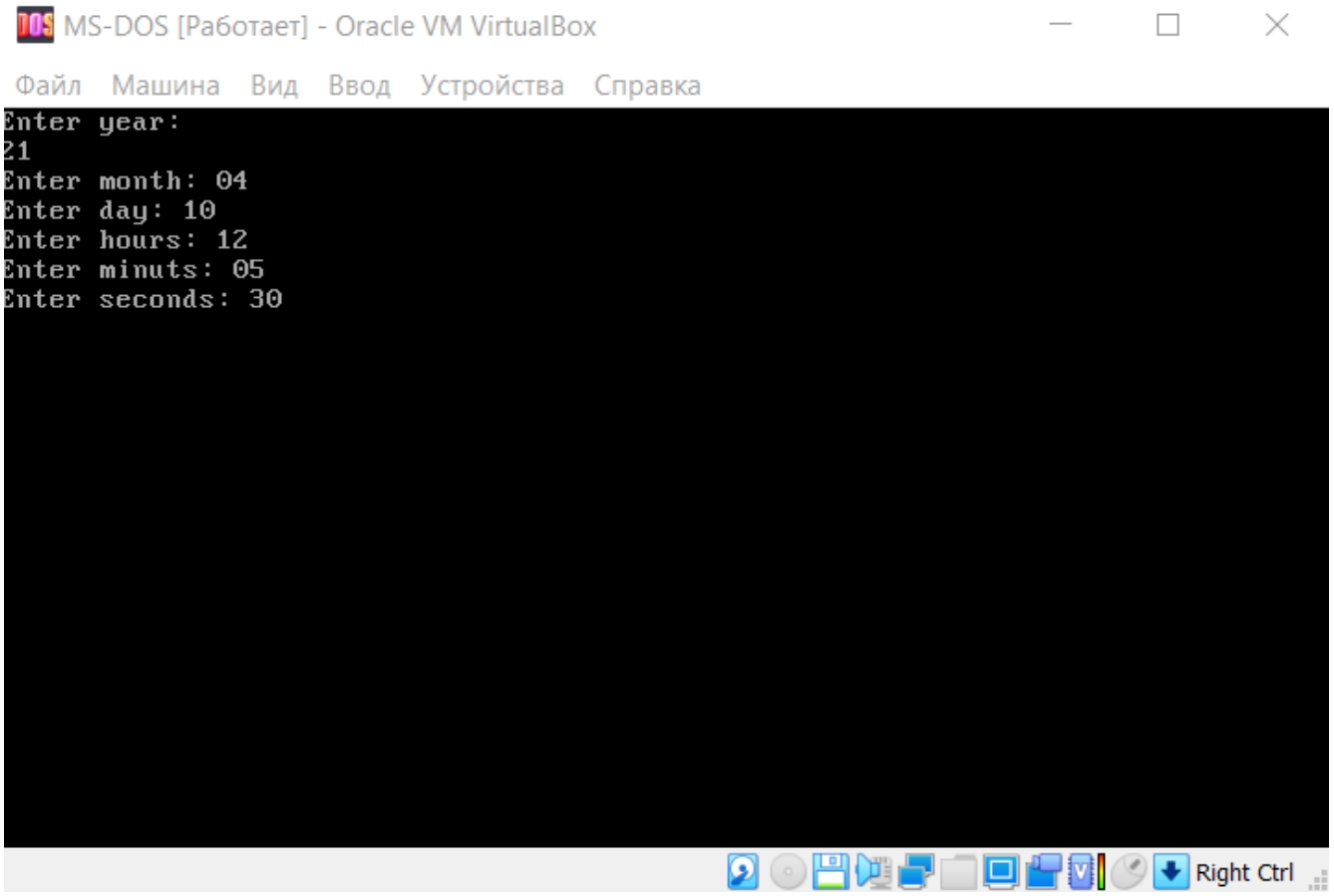

Рисунок 4.2 — Установка нового времени

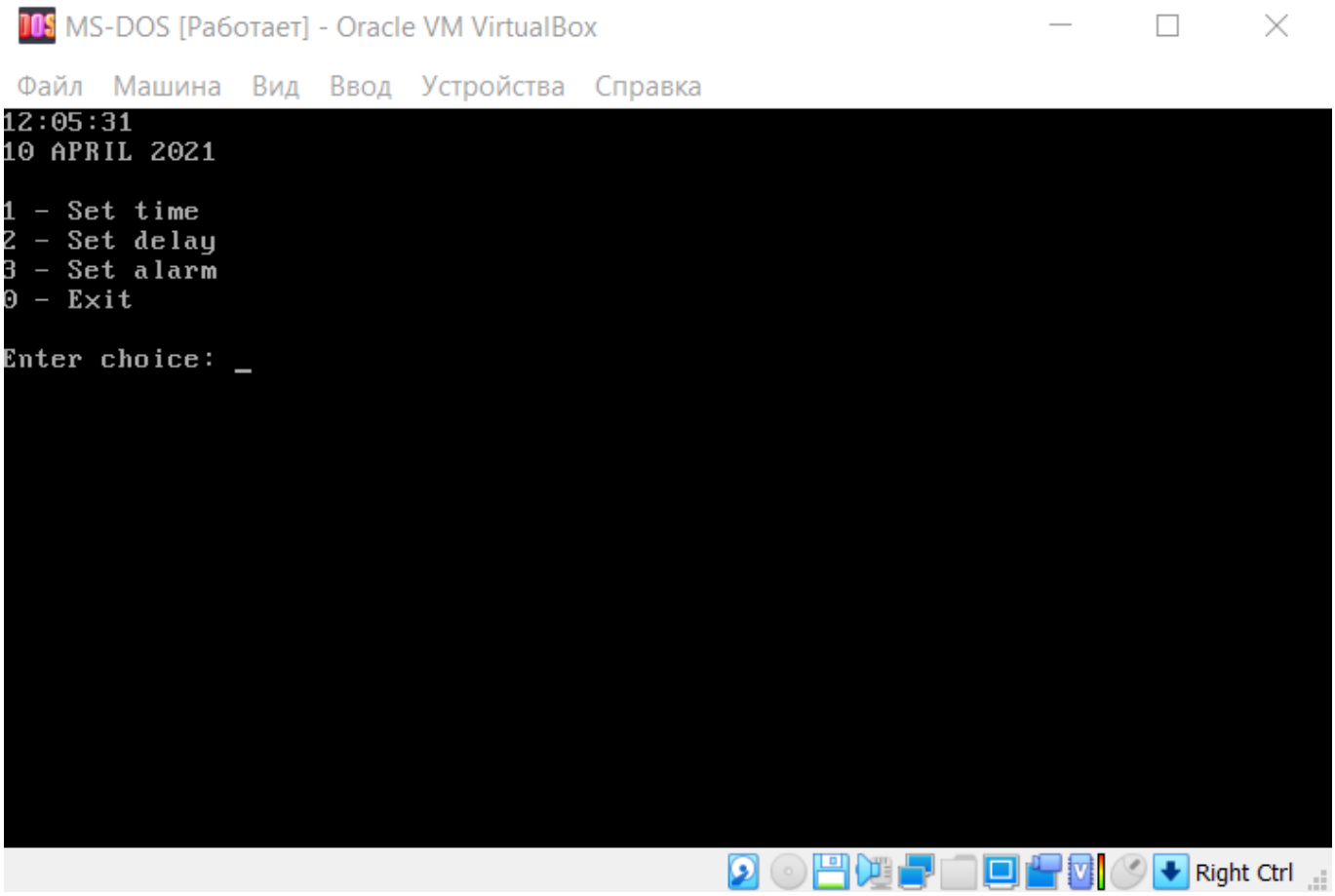

Рисунок 4.3 — Вывод нового времени.

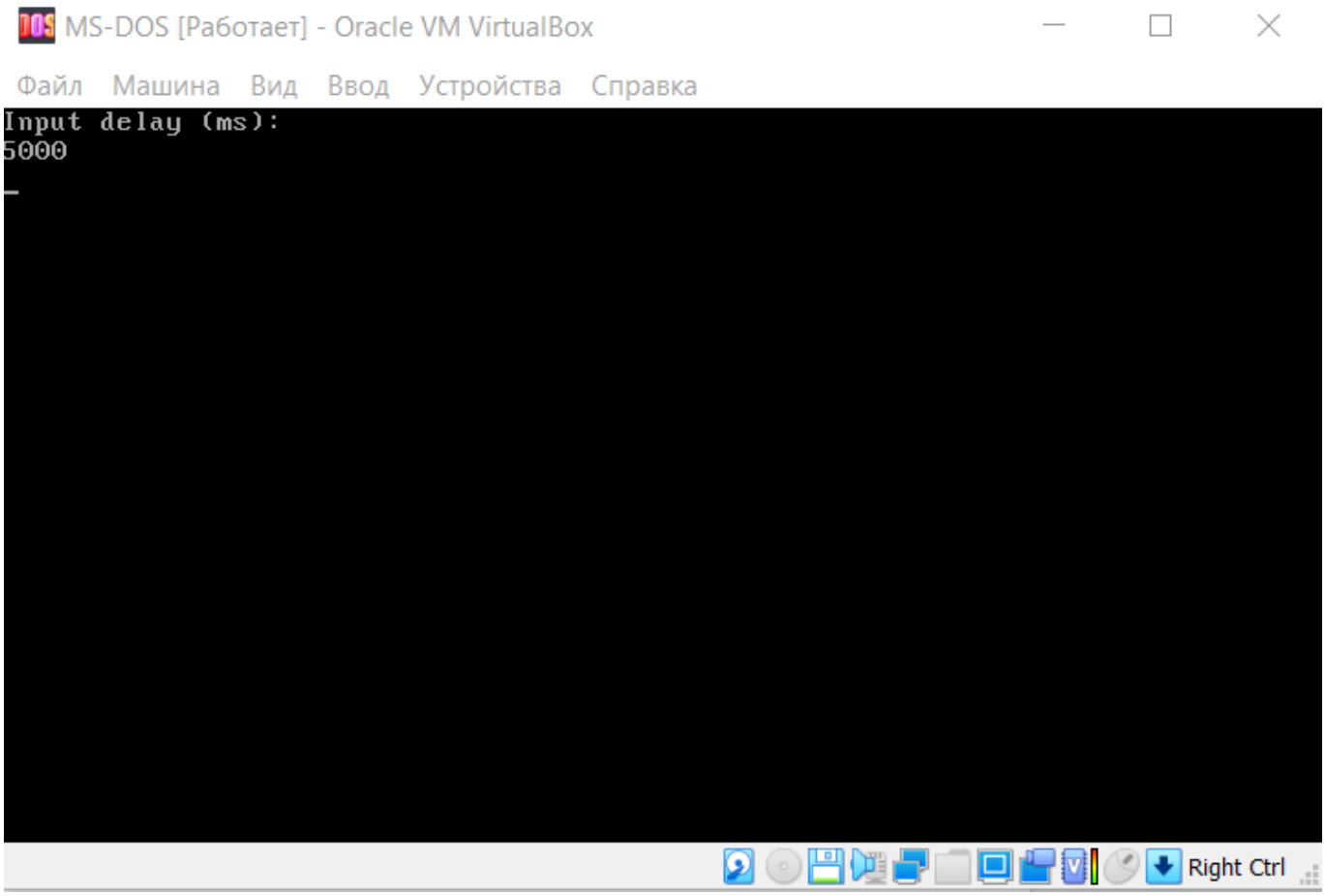

Рисунок 4.4 — Установка задержки.

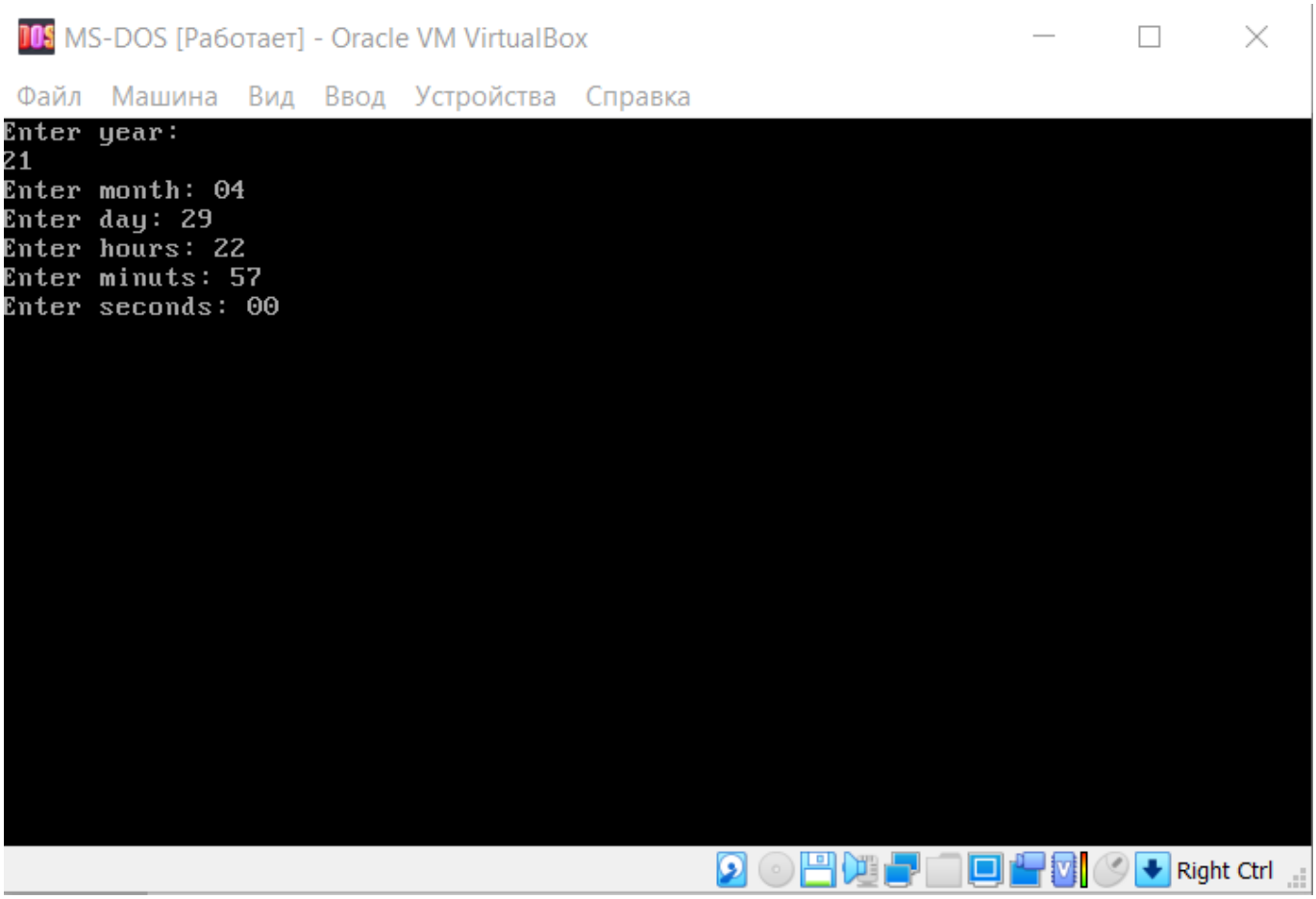

Рисунок 4.5 — Установка будильника.

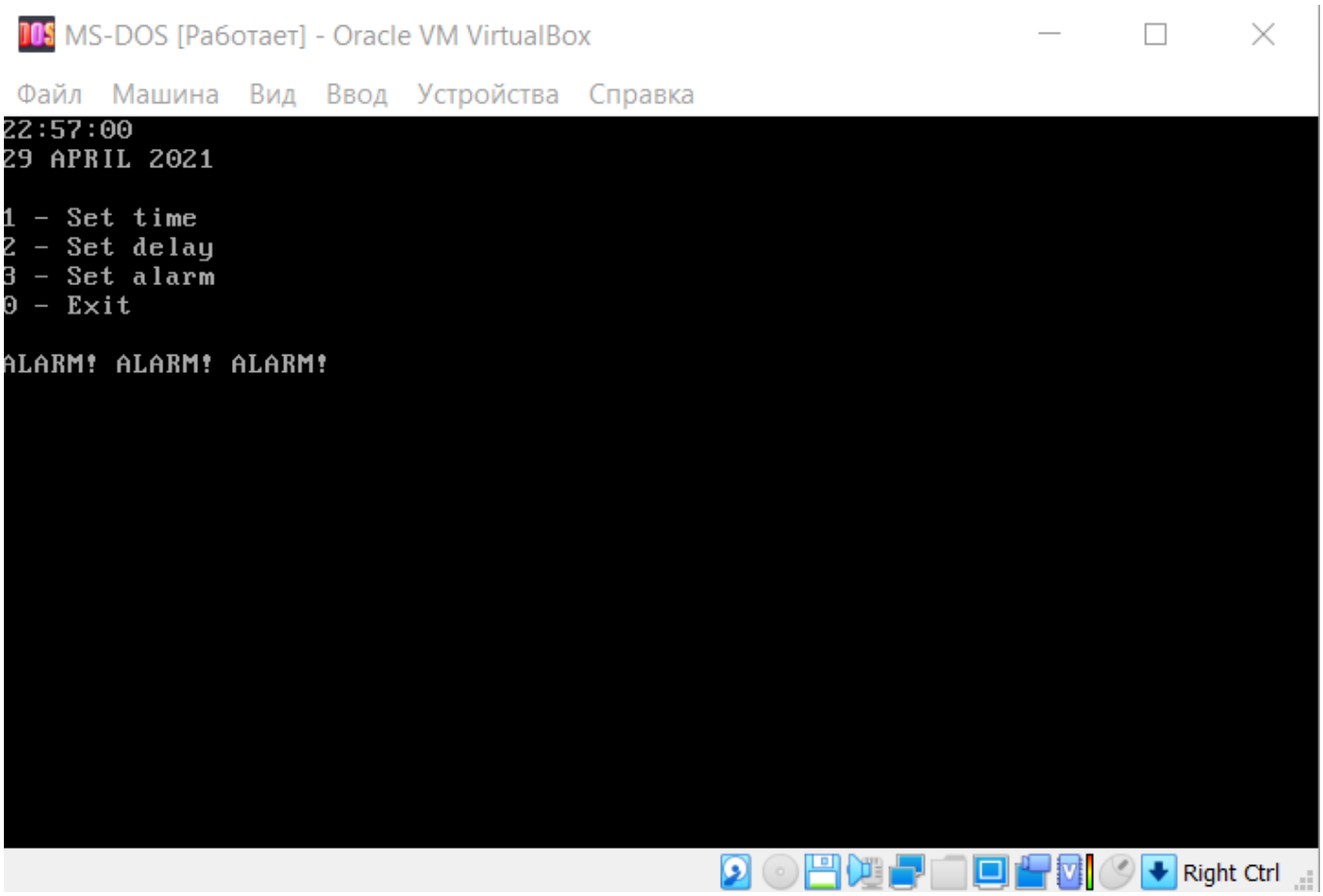

Рисунок 4.6 — Срабатывание будильника.

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.