Министерство образования Республики Беларусь

Учреждение образования Белорусский государственный университет информатики и радиоэлектроники

Кафедра ЭВМ

Отчёт по лабораторной работе №4

“Программирование часов реального времени”

Проверил: Выполнил:

Минск 2023

#### **Постановка задачи**

Задание по лабораторной работе состоит из двух частей. Первая часть общая для всех, вторая – по вариантам, выдаваемым преподавателем. К тому же имеются общие условия для всей лабораторной работы.

***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 (два варианта)***.

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. Задержка должна вводится с клавиатуры в миллисекундах в удобной для пользователя форме.
  2. После окончания периода задержки программа должна сообщить об этом в любой форме. При этом зависание компьютера не допускается.

1. Используя аппаратное прерывания часов реального времени и режим будильника ЧРВ (4A использовать не желательно) реализовать функции программируемого будильника.
   1. Время будильника вводится с клавиатуры в удобной для пользователя форме.
   2. При срабатывания будильника программа должна сообщить об этом в любой форме. При этом зависание компьютера не допускается.

***Общие условия для всей лабораторной работы.***

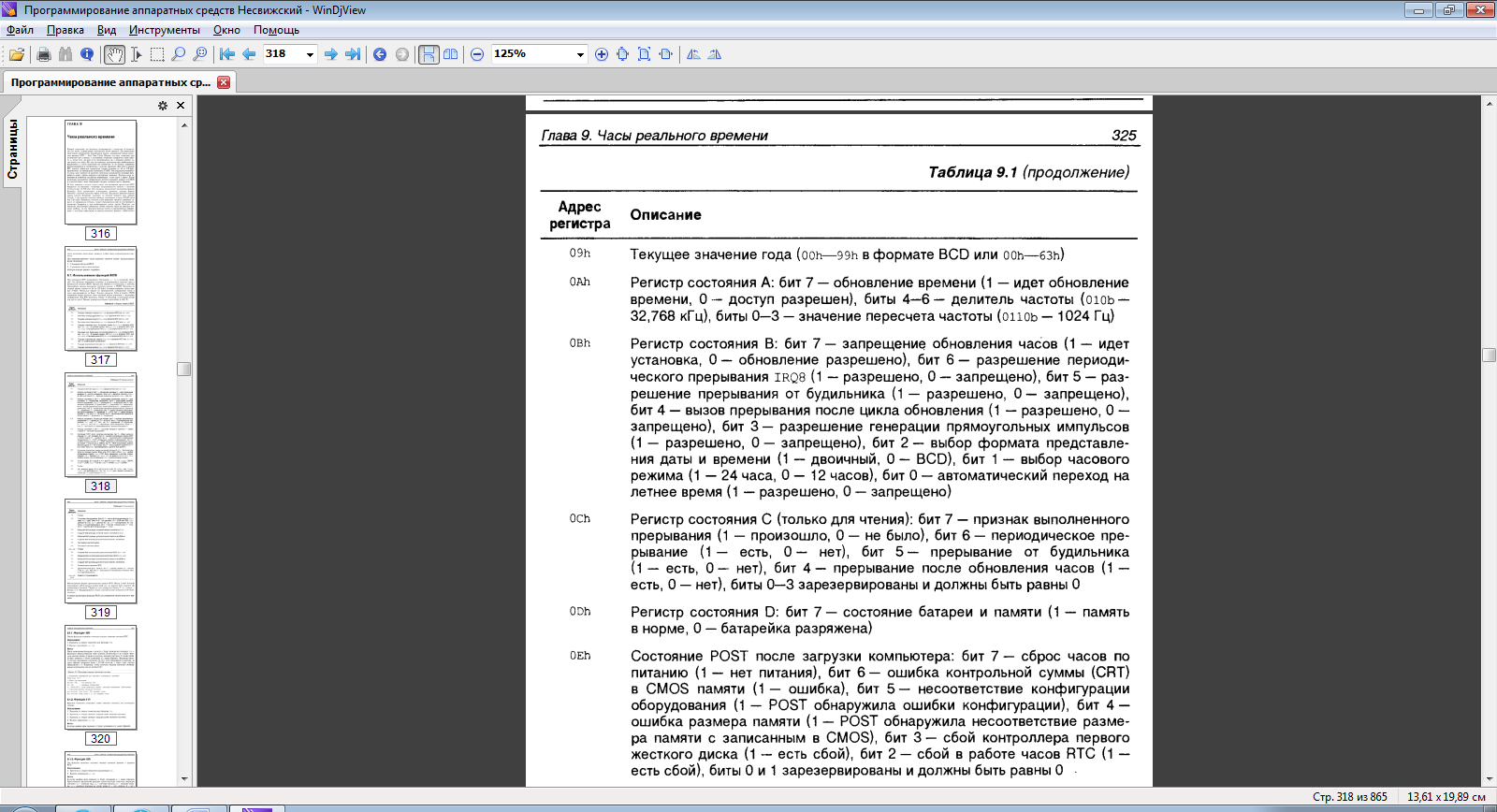
1. В программе должно быть реализовано меню, позволяющее выбрать тестируемый функционал (установка времени, считывание времени, задержка и т.д.)
2. Весь ввод/вывод данных с/на консоль должен выполняться в удобной для пользователя форме.
3. Зависание компьютера не допускается ни в ходе работы программы, ни после ее завершения.

#### **Алгоритм решения задачи**

Часы реального времени (Real Time Clock) - специальный модуль, используемый для непрерывного отсчёта времени. За счёт использования батарейки этот модуль работает даже когда компьютер выключен.

Для хранения данных в часах реального времени предусмотрено несколько регистров (данные записаны в формате BCD):

|  |  |
| --- | --- |
| **Адрес регистра** | **Описание значения** |
| 00h | Текущая секунда |
| 01h | Секунды будильника |
| 02h | Текущая минута |
| 03h | Минуты будильника |
| 04h | Текущий час |
| 05h | Часы будильника |
| 06h | Текущий день недели *(01-07, 01 - воскресенье)* |
| 07h | Текущий день месяца |
| 08h | Текущий месяц |
| 09h | Текущий год (последние 2 цифры) |



Для доступа к часам реального времени используются всего два порта: 70h и 71h. Порт 70h используется только для записи и позволяет выбрать адрес регистра в CMOS памяти. Порт 71h применяется как для записи, так и для чтения данных из указанного (через порт 70h) регистра CMOS. Оба порта являются 8-разрядными. Кроме того, бит 7 в порту 70h не относится к работе с RTC(ЧРВ), а управляет режимом немаскируемых прерываний (1 — прерывания запрещены). Если у вас будут проблемы с доступом к регистрам CMOS, следует запрещать прерывания (команда cli) перед началом работы и разрешать прерывания (команда sti) после. Но, как правило, этого не требуетcя.

Основное назначение часов - это непрерывный отсчет времени cуток даже в тот период, когда компьютер выключен, а также извещение процессора о наступлении определенного времени (режим будильника). Часы реального времени реализованы на базе микросхемы типа Motorola МС 146818 (RT/CMOS RAM Chip) и обеспечивают непрерывный отсчет времени при выключении питания на компьютере за счет аккумуляторной батареи. Эта микросхема включает также память, содержимое которой сохраняется при исключении питания. Аппаратура часов реального времени использует только первые 14 байт этой памяти, а остальные предназначены для сохранения конфигурации компьютера. Часы позволяют генерировать три типа прерываний по линии 0 второго контроллера прерываний (IRQ 8), что соответствует в IBM PC прерыванию 70h:

1. периодические прерывания с интервалом 976.562 мкс;
2. прерывание от будильника;
3. прерывание по окончанию обновления значения часов.

### Регистры часов

Как уже говорилось ранее, часы используют первые 14 байт памяти микросхемы типа Motorola MC146818, которые будем в дальнейшем называть регистрами часов. Чтобы считать любой памяти микросхемы, в том числе и данные из регистра часов необходимо выполнить следующие шаги:

1. записать адрес считываемого регистра в порт 70h;
2. считать данные регистра из порта 71h.

Например, следующие команды используются для чтения регистра 0 часов реального времени:

mov al,0 ; Записать адрес регистра

out 70h,al

jmp short $+2

in al,71h ; Считать регистр

Аналогично, для записи значения в регистр часов требуется занести адрес модифицируемого регистра в порт 70h, после чего вывести в порт 71h записываемое значение. Например:

mov al,0 ; Записать адрес регистра

out 70h,al

jmp short $+2

mov al,20h ; Занести новое значение

out 71h,al

Следует отметить, что чтение и запись данных порта 71h должны выполняться немедленно после записи номера регистра в порт 70h. Кроме того, в процессе операций ввода-вывода с регистрами часов должны быть запрещены прерывания, что позволит избежать модификации порта 70h процедурой обработки прерывания до чтения или записи данных.  
Адресация регистров часов приведена в таблице:

|  |  |
| --- | --- |
| Адрес регистра | Назначение регистра |
| 00h | Секунды |
| 01h | Секунды для будильника |
| 02h | Минуты |
| 03h | Минуты для будильника |
| 04h | Часы |
| 05h | Часы для будильника |
| 06h | День недели |
| 07h | Дата |
| 08h | Месяц |
| 09h | Год |
| 0Ah | Регистр состояния 1 |
| 0Bh | Регистр состояния 2 |
| 0Ch | Регистр состояния 3 |
| 0Dh | Регистр состояния 4 |
| 32h | Столетие |

Регистры времени 00h-09h и 32h содержат время в двоично-десятичном формате (например, дата 26 хранится в виде 26h). Столетие задастся в виде первых двух цифр полного номера года (например, 19h). На машинах PS/2 байт столетия расположен по адресу 37h.  
**Регистр состояния 1** (А)имеет следующий формат:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| UIP | DV | | | RS | | | |

Бит *UIP* определяет момент обновления показаний часов и может быть установлен в 0 - часы доступны для чтения или - 1 - аппаратура выполняет обновление показаний часов. Для правильного чтения или записи значений часов эти операции необходимо выполнять, когда значение бита UIP равно 0.  
Биты *DV* задают значение частоты для обновления показаний часов. Для правильного функционирования часов данные биты должны быть установлены в значение 010, что соответствует частоте 32.768 Кгц. Это единственная частота, обеспечивающая правильное время в часах.  
Биты *RS* позволяют выбрать делитель выходной частоты. Для правильного функционирования эти биты должны быть усыновлены в значение 0110, что соответствует выходной частоте прямоугольных колебаний 1.024 Кгц и интервалу времени между периодическими прерываниями 976.562 мкс.  
Формат **регистра состояния 2 (B)** приведен ниже:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| UPD | PIE | AIE | UIE | SQWE | DM | CF | DSE |

Бит *UPD* используется для обновления показания часов. Когда он устанавливается в 1, внутренний цикл обновления часов прекращается, и программа может инициализировать любой регистр часов. Для запуска часов надо установить этот бит в 0.   
Бит *PIE* управляет генерацией периодических прерываний t интервалом 976.562 мкс. Когда он установлен в 0, прерывания запрещены. При установке этого бита в 1, периодические прерывания разрешены.   
Аналогично, бит *AIE* задает, разрешены ли прерывания от будильника, а бит *UIE* разрешает или запрещает прерывания при завершении цикла обновления показаний часов.  
Бит *SQWE* разрешает или запрещает генерацию прямоугольных колебаний (см. биты RS в регистре состояния 1). Функция генерации прямоугольных колебаний не используется в компьютере и бит устанавливается в 0.   
*DM* определяет формат времени и даты. Если этот бит установлен в 0, то используется двоично-десятичный формат. При единичном значении бита данные представляются в двоичном виде. Для IBM PC этот бит установлен в 0.  
Бит *CF* задаст формат представления времени. Значение О указывает, что используется 12-ти часовой формат, а значение 1 - 24-х часовой формат. В компьютерах IBM PC используется 24-х часовой формат.  
Бит *DSE* установлен в 1, если в часы заносится летнее время и программы, пользующиеся показаниями часов, осуществляют пересчет времени. Обычно эта функция не используется, и бит установлен в 0.  
**Регистр состояния 3(C)** доступен только по чтению. Он содержит текущий статус прерывания от часов и имеет следующий формат:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INT | PI | AI | UI | Резерв | | | |

Бит *INT* указывает на наличие прерывания (0 - нет прерывания, 1 - выработано прерывание).  
Биты *PI, АI и UI* уточняют тип прерывания, выработанного часами. Единичное значение бита PI указывает, что выработано периодическое прерывание. Аналогично, единичное значение бита AI сигнализирует о генерации прерывания от будильника, а бита UI - прерывания об окончании обновления показаний часов.   
Обработчик прерывания по линии IRQ8 должен прочитать регистр состояния 3. Иначе, следующее прерывание не будет выработано.  
**Регистр состояния 4** доступен только по чтению и имеет следующий формат:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VRB | Резерв | | | | | | |

В этом регистре состояния задействован только бит *VRB* который сигнализирует о состоянии аккумуляторной батареи. Единичное значение говорит о нормальном питании. Нулевое значение указывает, что батарея разряжена и данные в часах недостоверны.

1. **Листинг программы**

#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[] =

{

"JAN",

"FEB",

"MAR",

"APR",

"MAY",

"JUN",

"JUL",

"AUG",

"SEP",

"OCT",

"NOV",

"DEC"

};

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 - time");

printf("\n2 - delay");

printf("\n3 - alarm");

printf("\n0 - exit");

if (alarmOn == 1) printf("\n\nALARM ON");

if (alarmOn == 2) {

printf("\n\nWake up!");

delay(5000);

alarmOn = 0;

}

printf("\n\nwrite 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("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 А

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 В

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("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("hours: ");

scanf("%i", &enter);

} while ((enter > 23 && enter < 0));

data[2] = ConvertToBCD(enter);

do {

rewind(stdin);

printf("minuts: ");

scanf("%i", &enter);

} while (enter > 59 && enter < 0);

data[1] = ConvertToBCD(enter);

do {

rewind(stdin);

printf("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 В

outp(0x71, inp(0x71) | 0x40); // 0x40 = 0100 0000

// 6-й бит регистра B установлен в 1 для периодического прерывания

delayTime = 0;

while (delayTime <= delayMs)

{

printf("Delay time: %d ms\n", delayTime);

if (kbhit()) // если нажата клавиша на клавиатуре

{

getch(); // считываем символ с клавиатуры

break; // выходим из цикла

}

}

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

lastAlarm = NULL;

}

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");

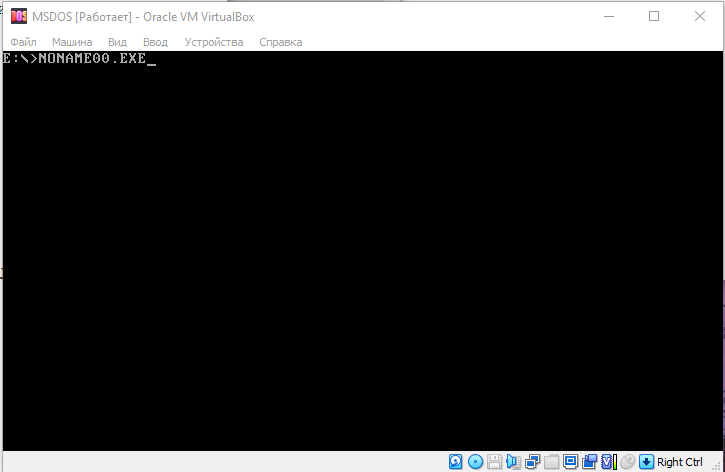
alarmOn = 2;

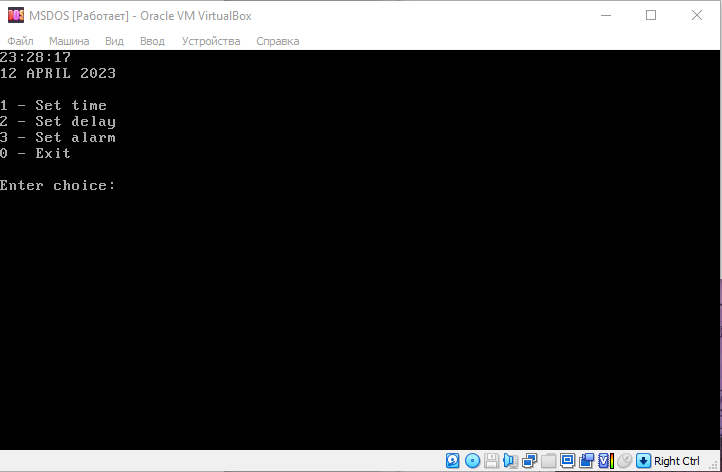
lastAlarm();

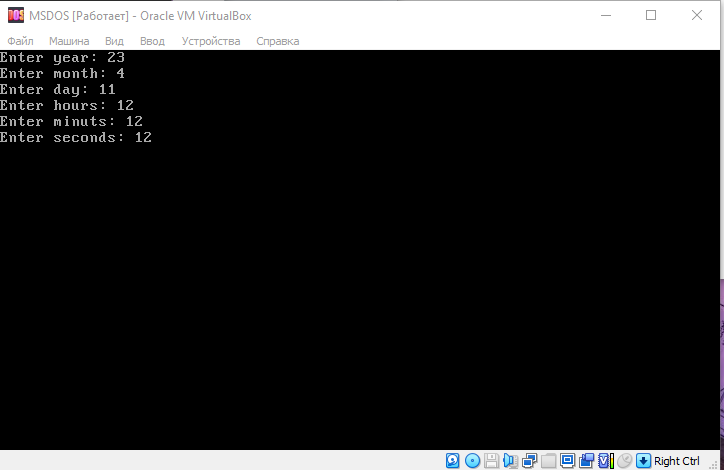
resetAlarm();

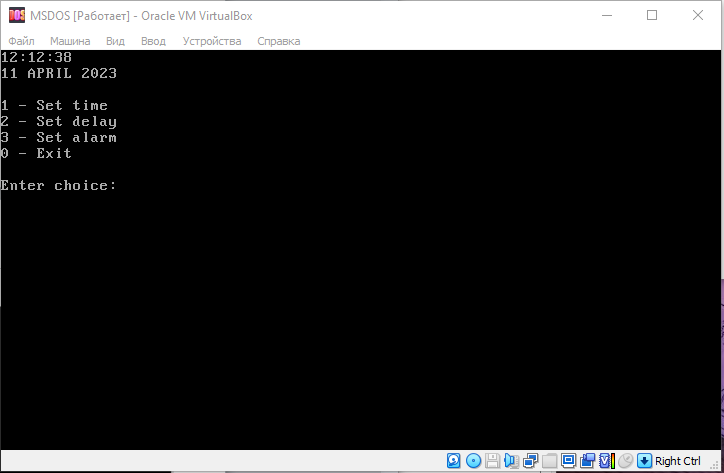
}

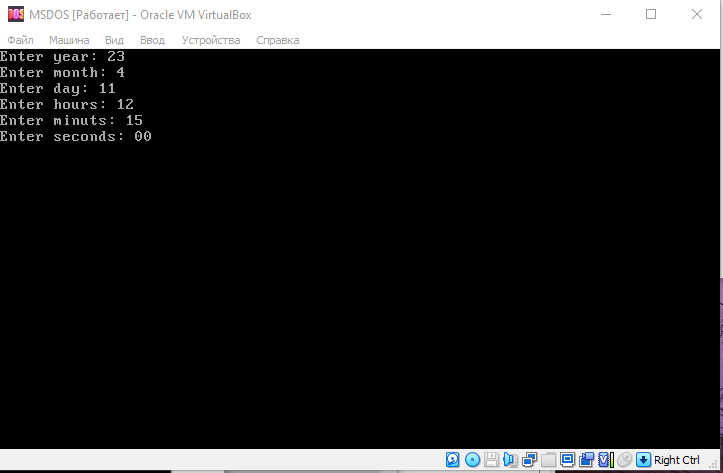
**Тест**

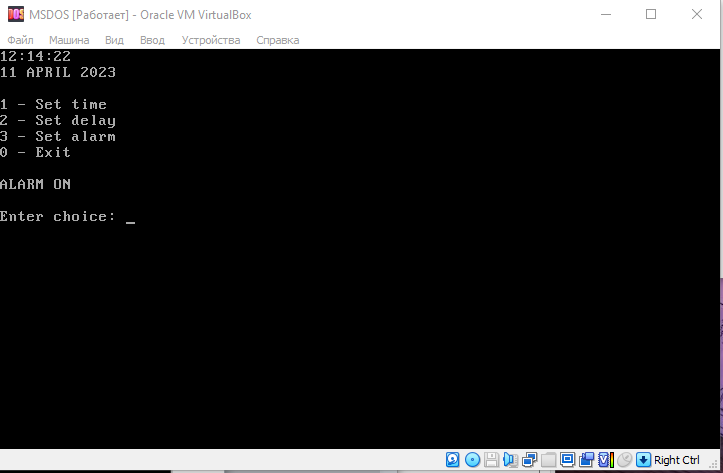


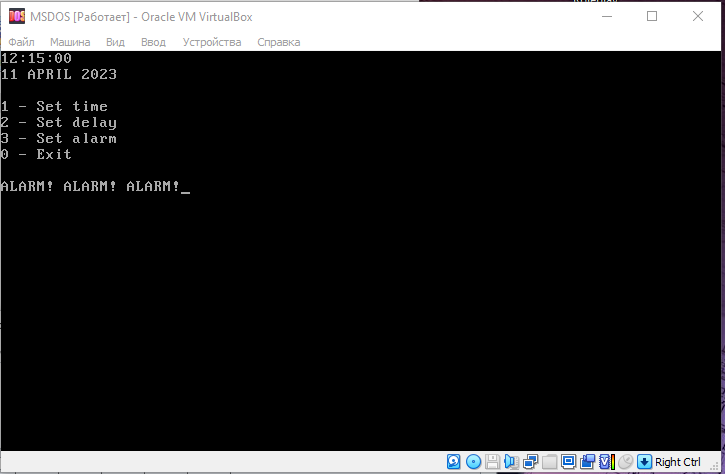






****

****

****

#### **Вывод**

Познакомился с часами реального времени и запрограммировал их на виртуальной машине. Написал программу, которая считывает и устанавливает время в часах реального времени. Используя аппаратное прерывание часов реального времени и режим генерации периодических прерываний реализовал функцию задержки с точностью в миллисекунды.