БГУИР
Кафедра ЭВМ

Отчет по лабораторной работе № 5

Тема: «Часы реального времени»

Выполнил:
студент группы 724402 Чернявский Я.А.

Проверил:
доцент каф. ЭВМ _____ Селезнёв И.Л.

Минск
2019

1. Цель работы

Получить навыки программирования и изучить возможности использования часов реального времени.

2. Задание

Под MS DOS написать программу, которая:

1) считывает и устанавливает время в часах реального времени;
2) реализует задержку с точностью до 1 миллисекунды;

3. Теоретические сведения

В состав IBM PC AT входят часы реального времени Real Time Clock (RTC), которые являются специальным модулем, используемым для непрерывного отсчёта времени, и 64 байта неразрушающейся оперативной КМОП (комплементарный металлооксидный полупроводник памяти) или CMOS (Complementary Metal-Oxide-Semiconductor), питающиеся от автономного источника питания.

CMOS-память – т.н. полупостоянная память, которая с одной стороны хранит информацию очень долго, что даёт основание считать её постоянной, но с другой стороны она всё же имеет, хоть и крайне малое, энергопотребление. CMOS-память применяется в первую очередь как часть памяти BIOS, которая хранит основную конфигурацию компьютера, даже когда последний выключен. При выключенном компьютере эта память работает от маломощной батарейки или аккумулятора. При включении ПК содержимое CMOS анализируется POST (Power-On Self-Test), который извлекает из нее конфигурацию системы и текущие дату и время. Часы реального времени RTC и CMOS память первоначально выполнялись на базе микросхемы MC14818 фирмы Motorola, а в современных компьютерах реализуется чипсетом.

Краткий перечень того, что хранится в регистрах CMOS BIOS:

- приоритет загрузки
- количество памяти
- режимы энергопотребления
- информация о периферийных устройствах
- дата/время и регистры будильника (ЧРВ)

Упрощенная структурная схема RTC представлена на рисунке 3.1.

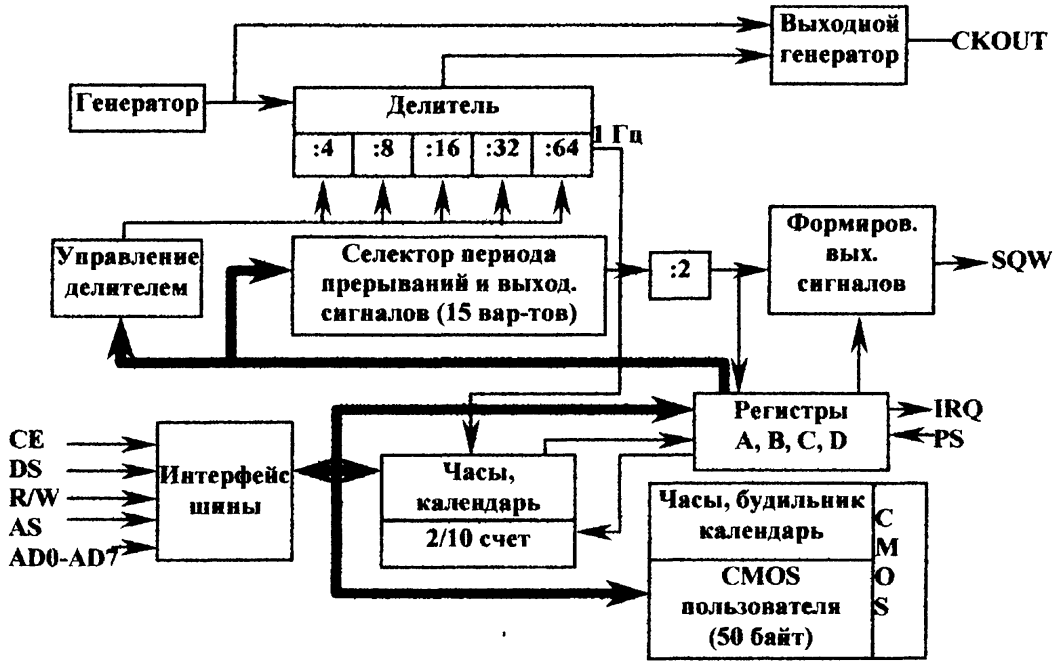

Рисунок 3.1 – Структурная схема RTC и CMOS памяти

Назначение сигналов следующее:

- CE – разрешение кристалла (выход дешифратора адреса);
- DS – строб данных;
- AS – адресный строб;
- R/W – чтение/запись;
- SQW – выходные прямоугольные импульсы – меандр;
- IRQ – запрос на прерывание от RTC;
- PS – сигнал состояния питания (контроль достоверности данных);
- CKOUT – синхросигнал

Основная часть RTC – задающий генератор с частотой 4.2 МГЦ, 1.04 МГц и делитель частоты, у которого выход последнего каскада (1 ГЦ) управляет часами.

Несмотря на полный размер CMOS памяти от 64 до 128 байт, стандартизированы только первые 51 байт. Остальные зависят от производителя материнской платы. Каждое смещение байта памяти CMOS определяет номер регистра, через который можно считывать и записывать информацию. Для RTC выделены первые 13 регистров, а остальные служат для других целей. Формат микросхемы памяти представлен на рисунках 3.2 – 3.4.

Адрес регистра	Описание
00h	Текущее значение секунды (00h—59h в формате BCD или 00h—3Bh)
01h	Значение секунд будильника (00h—59h в формате BCD или 00h—3Bh)
02h	Текущее значение минуты (00h—59h в формате BCD или 00h—3Bh)
03h	Значение минут будильника (00h—59h в формате BCD или 00h—3Bh)
04h	Текущее значение часа: 24-часовой режим (00h—23h в формате BCD или 00h—17h), 12-часовой режим AM (01h—12h в формате BCD или 00h—0Ch), 12-часовой режим PM (81h—92h в формате BCD или 81h—8Ch)
05h	Значение часа будильника: 24-часовой режим (00h—23h в формате BCD или 00h—17h), 12-часовой режим AM (01h—12h в формате BCD или 00h—0Ch), 12-часовой режим PM (81h—92h в формате BCD или 81h—8Ch)
06h	Текущее значение дня недели (01h—07h в формате BCD или 01h—07h, где 01h соответствует воскресенью)
07h	Текущее значение дня месяца (01h—31h в формате BCD или 01h—1Fh)
08h	Текущее значение месяца (01h—12h в формате BCD или 01h—0Ch)

Рисунок 3.2 – Формат памяти CMOS (часть 1)

Адрес регистра	Описание
09h	Текущее значение года (00h—99h в формате BCD или 00h—63h)
0Ah	Регистр состояния A: бит 7 — обновление времени (1 — идет обновление времени, 0 — доступ разрешен), биты 4—6 — делитель частоты (010b — 32,768 кГц), биты 0—3 — значение пересчета частоты (0110b — 1024 Гц)
0Bh	Регистр состояния B: бит 7 — запрещение обновления часов (1 — идет установка, 0 — обновление разрешено), бит 6 — разрешение периодического прерывания IRQ8 (1 — разрешено, 0 — запрещено), бит 5 — разрешение прерывания от будильника (1 — разрешено, 0 — запрещено), бит 4 — вызов прерывания после цикла обновления (1 — разрешено, 0 — запрещено), бит 3 — разрешение генерации прямоугольных импульсов (1 — разрешено, 0 — запрещено), бит 2 — выбор формата представления даты и времени (1 — двоичный, 0 — BCD), бит 1 — выбор часового режима (1 — 24 часа, 0 — 12 часов), бит 0 — автоматический переход на летнее время (1 — разрешено, 0 — запрещено)
0Ch	Регистр состояния C (только для чтения): бит 7 — признак выполненного прерывания (1 — произошло, 0 — не было), бит 6 — периодическое прерывание (1 — есть, 0 — нет), бит 5 — прерывание от будильника (1 — есть, 0 — нет), бит 4 — прерывание после обновления часов (1 — есть, 0 — нет), биты 0—3 зарезервированы и должны быть равны 0
0Dh	Регистр состояния D: бит 7 — состояние батареи и памяти (1 — память в норме, 0 — батарейка разряжена)
0Eh	Состояние POST после загрузки компьютера: бит 7 — сброс часов по питанию (1 — нет питания), бит 6 — ошибка контрольной суммы (CRT) в CMOS памяти (1 — ошибка), бит 5 — несоответствие конфигурации оборудования (1 — POST обнаружила ошибки конфигурации), бит 4 — ошибка размера памяти (1 — POST обнаружила несоответствие размера памяти с записанным в CMOS), бит 3 — сбой контроллера первого жесткого диска (1 — есть сбой), бит 2 — сбой в работе часов RTC (1 — есть сбой), биты 0 и 1 зарезервированы и должны быть равны 0
0Fh	Состояние компьютера перед последней загрузкой: 00h — был выполнен сброс по питанию (кнопка Reset или <Ctrl>+<Alt>+<Del>), 03h — ошибка тестирования памяти, 04h — POST была завершена и система перезагружена, 05h — переход (jmp dword ptr) на адрес в 0040h:0067h, 07h — ошибка защиты при тестировании, 08h — ошибка размера памяти
10h	Тип дисковода (0—3 для A и 4—7 для B): 0000b — нет, 0001b — 360 Кб, 0010b — 1,2 Мб, 0011b — 720 Кб, 0100b — 1,44 Мб, 0101b — 2,88 Мб
11h	Резерв
12h	Тип жесткого диска (0—3 для D и 4—7 для C): 0000b — нет, 0001b — 1110b — тип дисковода (от 1 до 14), 1111b — диск первый описывается в регистре 19h, а диск второй — в 1Ah

Рисунок 3.3 – Формат памяти CMOS (часть 2)

Адрес регистра	Описание
13h	Резерв
14h	Состояние оборудования: биты 6—7 — число флоппи-дисководов (00b — один, 01b — два), биты 4—5 — тип дисплея (00b — EGA или VGA, 01b — цветной 40 × 25, 10b — цветной 80 × 25, 11b — монохромный 80 × 25), биты 2 и 3 зарезервированы, бит 1 — наличие сопроцессора (1 — есть), бит 0 — наличие флоппи-дисковода (1 — есть)
15h	Младший байт размера основной памяти в килобайтах (80h)
16h	Старший байт размера основной памяти в килобайтах (02h)
17h	Младший байт размера дополнительной памяти в килобайтах
18h	Старший байт размера дополнительной памяти в килобайтах
19h	Тип первого жесткого диска
1Ah	Тип второго жесткого диска
1Bh—2Dh	Резерв
2Eh	Старший байт контрольной суммы регистров CMOS (10h—2Dh)
2Fh	Младший байт контрольной суммы регистров CMOS (10h—2Dh)
30h	Младший байт размера дополнительной памяти в килобайтах
31h	Старший байт размера дополнительной памяти в килобайтах
32h	Значение века в формате BCD
33h	Дополнительный флаг свойств: бит 7 — размер памяти (1 — больше 1 Мб, 0 — до 1 Мб), бит 6 — используется программой установки, биты 0—5 зарезервированы
34h—3Fh (7Fh)	Зависят от производителя

Рисунок 3.4 – Формат памяти CMOS (часть 3)

Используемый формат представления данных BCD (Binary Coded Decimal) представляет собой двоично-десятичный код, где каждый байт содержит два независимых значения и каждый десятичный разряд числа записывается в виде его четырёхбитного двоичного кода. Первое из значений кодируется битами 7 – 4, а второе – битами 3 – 0. Поддерживаются только положительные числа до 99 включительно.

Формат регистров состояния RTC A, B и C, D приведены на рисунках 3.5 и 3.6 соответственно.

0Ah - Регистр A состояния RTC:

7	6	5	4	3	2	1	0
UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0

Биты:

0-3: (RS0-RS3) скорость отсчета (равна 0110);

4-6: (DV0-DV2) делитель (равен 010);

7: (UIP) флаг обновления (0 - можно читать данные RTC, 1 - идет корректировка).

0Bh- Регистр B состояния RTC:

7	6	5	4	3	2	1	0
SET	PIE	AIE	UIE	SQWE	DM	24/12	DSE

Биты:

0: (DSE) разрешение летнего времени; 1 -разрешено, 0 - запрещено (равно 0);

1: 12- или 24-часовое время (равен 1 - 24 часа);

2: (DM) формат данных: 0 - BCD, 1 - двоичный (равен 0);

3: (SQWE) 1 - разрешить прямоугольный импульс (равен 0);

4: (UIE) 1 - разрешить прерывание по концу обновления - каждую секунду (равен 0);

5: (AIE) 1 - разрешить сигнальное прерывание (равен 0);

6: (PIE) 1 - разрешить периодические прерывания с частотой RS3-RS0 регистра A;

7: (SET) запрещение корректировки; 1 - нет коррекции, 0 - есть коррекция (равен 0).

Рисунок 3.5 – Формат регистров состояния A и B

0Ch- Регистр C состояния RTC: биты состояния прерываний, только для чтения. При чтении из регистра все разряды сбрасываются.

7	6	5	4	3	2	1	0
IRQF	PF	AF	UF	0	0	0	0

Биты:

0-3: резерв;

4-6: причины запроса:

6: (PF) периодическое прерывание;

5: (AF) сигнальное прерывание (текущее время совпало с временем будильника);

4: (UF) прерывание по концу корректировки;

7: (IRQF) запрос на прерывание.

0Dh - Регистр D состояния RTC. Бит 7=1, если CMOS получает питание; 0=нет питания от автономного источника.

Рисунок 3.6 – Формат регистров состояния C и D

Часы реального времени вырабатывают аппаратное прерывание IRQ8, которому соответствует прерывание с номером 70h. Это прерывание может вырабатываться по трем причинам:

- Прерывание по окончанию изменения данных. Вырабатывается при установленном в 1 бите 4 регистра состояния B после каждого обновления регистров часов.
- Прерывание будильника вырабатывается при совпадении регистров часов и будильника и при установленном в 1 бите 5 регистра состояния B.
- Периодическое прерывание вырабатывается с интервалом примерно 1 миллисекунда при установленном в 1 бите 6 регистра состояний B.

При срабатывании будильника BIOS вырабатывает прерывание INT 4Ah.

Использование портов

Для доступа к часам реального времени используются 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, а управляет режимом немаскируемых прерываний (1 – прерывания запрещены).

Для того, чтобы считать время из часов, требуется выбрать один из регистров (00 – 09) через порт 70h и считать значение указанного регистра через порт 71h (outp(0x70, reg); res = inp(0x71);).

Перед началом установки новых значений времени необходимо считывать и анализировать старший бит регистра состояния A 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 (do{ outp(0x70, 0x0A);} while (inp(0x71) & 0x80);).

После проверки доступности регистров, необходимо отключить внутренний цикл обновления часов реального времени, воспользовавшись старшим битом регистра состояния B (outp(0x70, 0x0B); outp(0x71, inp(0x71) | 0x80). После операции установки значений этот бит должен быть сброшен, для возобновления внутреннего цикла обновления часов реального времени (outp(0x70, 0x0B); outp(0x71, inp(0x71) & 0x7F);).

Для реализации задержки времени требуется установить свой обработчик прерывания RTC, размаскировать линии сигнала запроса RTC (младший бит A1 установить в «0» (outp(0xA1, inp(0xA1) & 0xFE);), включить периодические

прерывания (бит 6 регистра B установить в 1) и ожидать указанное количество милисекунд. После задержки требуется восстановить старый обработчик.

4. Листинг кода программы

```
// 724402-2 Chernyavsky Y.A.
// Laboratornaya rabota 5 "Chasi realnogo vremeni"

#include <dos.h>
#include <ctype.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
#pragma warn -sus

int msCounter = 0;  // Schetchik dlya zaderzhki
int x = 45;                 // Global'nye peremennye stroki i stolbca
int y = 7;                  //  dlya vyvoda na ehkran
char massive[20];    //  massiv dlya vvoda dannyh
int nomer = 0;              // kolichestvo ehlementov v massive
int enter = 0;              // peremennaya dlya klavishi enter
int gran = 0;        // granica dlya backspace
int tik = 0;         //  Schetchik vyzova preryvaniya tajmera

void interrupt far (*oldInt70h)(void);          // Ukazatel' na funkciyu obrabotki
preryvanij RTC
void interrupt far NewInt70Handler(void); // Funkciya obrabotki preryvanij RTC


void CleanScreen(int line, int counts, int column);  // Funkciya ochistki ehkrana
void LockClockUpdate(void);              // Zapret na obnovlenie tajmera
void UnlockClockUpdate(void);            // Razreshenie na obnovlenie tajmera
int BCDToInteger(int bcd);               // Perevod v tip int
unsigned char IntToBCD(int value);  // Perevod v dvoichno-desyatichnyj kod
void GetTime(void);
void SetTime(void);
void CustomDelay(void);           // Funkciya zaderzhki vremeni
void redakttime(void);            // Funkciya redaktirovaniya vremeni
void WaitClockIsFree(void); // Provekrka na vozmozhnost' obrashcheniya k chasam

void interrupt newInt9(void);            // Funkciya obrabotki preryvaniya klaviatury
void interrupt(*oldInt9)(void);          // Ukazatel' na obrabotchik preryvaniya klaviatury
void Vprint(char *str);                        // Funkciya vyvoda
void StaticLines(char *str, int line, int column); // Funkciya vyvoda
void inittimer(void);                          // Inicializaciya tajmera
void inittimerstop(void);                // Zamena tajmera na standartnyj
void cleanmassive(void);                 // Funkciya ochistki massiva

void interrupt TimerHandler(void);       // Funkciya obrabotki preryvanij sistemnogo tajmera
void interrupt(*oldTimerHandler)(void); // Ukazatel' na funkciyu obrabotki preryvanij
sistemnogo tajmera

void main()
{
       char c;
       clrscr();
       StaticLines("724402-2 Chernyavsky Y.A. Laboratornaya rabota 5 'Chasi realnogo
vremeni'", 0, 0);
```

```
        StaticLines("Press 1 - Show time", 1, 0);
        StaticLines("Press 2 - Set time", 2, 0);
        StaticLines("Press 3 - Delay time", 3, 0);
        StaticLines("Press Esc - quit", 4, 0);
        oldTimerHandler = getvect(0x1c);

        while(c != 27)
        {
                c = getch();
                switch(c)
                {
                        case '1':
                                inittimer();
                                GetTime();
                                StaticLines("Input  to continue", 8, 0);
                                getch();
                                CleanScreen(6, 720, 0);
                                inittimerstop();
                                break;
                        case '2':
                                inittimer();
                                SetTime();
                                break;
                        case '3':
                                inittimer();
                                CustomDelay();
                                break;
                        case 27:
                                inittimerstop();
                                return;
                }
                c = 0;
        }


}
void inittimer() {
        disable();                      // Preryvanie tajmera
        setvect(0x1c, TimerHandler);    // возникает 18,2 раза в секунду
        enable();
}
void inittimerstop() {
        disable();
        setvect(0x1c, oldTimerHandler);   // Установка старого обработчика
        enable();
}
void interrupt newInt9()    // Функция обработки прерывания клавиатуры
{
        char str6[9];
        unsigned char value = 0;
        value = inp(0x60);    // Poluchenie znacheniya iz porta 60h
        if (value == 0x03)   // if 2
        {
                itoa(2, str6, 10);
                Vprint(str6);
                massive[nomer] = '2';
                nomer++;
        }
        if (value == 0x02)  // if 1
        {
```

```
        itoa(1, str6, 10);
        Vprint(str6);
        massive[nomer] = '1';
        nomer++;
}
if (value == 0x04)   // if 3
{
        itoa(3, str6, 10);
        Vprint(str6);
        massive[nomer] = '3';
        nomer++;
}
if (value == 0x05)   // if 4
{
        itoa(4, str6, 10);
        Vprint(str6);
        massive[nomer] = '4';
        nomer++;
}
if (value == 0x06)   // if 5
{
        itoa(5, str6, 10);
        Vprint(str6);
        massive[nomer] = '5';
        nomer++;
}
if (value == 0x07)   // if 6
{
        itoa(6, str6, 10);
        Vprint(str6);
        massive[nomer] = '6';
        nomer++;
}
if (value == 0x08)   // if 7
{
        itoa(7, str6, 10);
        Vprint(str6);
        massive[nomer] = '7';
        nomer++;
}
if (value == 0x09)   // if 8
{
        itoa(8, str6, 10);
        Vprint(str6);
        massive[nomer] = '8';
        nomer++;
}
if (value == 0x0A)   // if 9
{
        itoa(9, str6, 10);
        Vprint(str6);
        massive[nomer] = '9';
        nomer++;
}
if (value == 0x0B)   // if 0
{
        itoa(0, str6, 10);
        Vprint(str6);
        massive[nomer] = '0';
        nomer++;
```

```
        }
        if (value == 0x0E)   // if backspace
        {
                if (x != gran) {
                        x--;
                        Vprint(" ");
                        x--;
                        massive[nomer] = '\0';
                        nomer--;
                        massive[nomer] = '\0';
                }
        }
        if (value == 0x1C)   // if Enter
        {
                enter = 1;
        }
        outp(0x20, 0x20);   // Sbros kontrollera preryvaniya
}

void WaitClockIsFree()
{
        do
        {
                outp(0x70, 0x0A);
        }
        while( inp(0x71) & 0x80 ); // proverka na 1 v 7 bite
}
void interrupt TimerHandler() // Funkciya obrabotki preryvaniya tajmera
{
        char str6[9];
        disable();
        tik++;              // Uvelichivaem kolichestvo tikov
        if (!(tik % 18))    // esli kratno 18,to proshla 1 sekunda
        {
                CleanScreen(6, 79, 0);
                GetTime();      // Vyvod vremeni i daty
        }
        outp(0x20, 0x20);  // Sbros kontrollera preryvaniya
        enable();
}
void GetTime()
{
        unsigned char value;
        char str6[9];
        StaticLines("Time", 6, 0);
        WaitClockIsFree(); // Proverka , svobodnyj li tajmer
        outp(0x70, 0x04); // Schityvanie dannyh dlya chasov
        value = inp(0x71);
        itoa((BCDToInteger(value)), str6, 10);
        if (BCDToInteger(value) < 10)
        {
                StaticLines("0", 6, 6);
                StaticLines(str6, 6, 7);
        }
        else {
                StaticLines(str6, 6, 6);
        }
        StaticLines(":", 6, 8);
        if (BCDToInteger(value) > 23) {
                redakttime();
```

```
        StaticLines("00", 6, 6);
}

WaitClockIsFree();
outp(0x70, 0x02); // Schityvanie dannyh dlya minut
value = inp(0x71);
itoa((BCDToInteger(value)), str6, 10);
if (BCDToInteger(value) < 10)
{
        StaticLines("0", 6, 9);
        StaticLines(str6, 6, 10);
}
else {
        StaticLines(str6, 6, 9);
}
StaticLines(":", 6, 11);

WaitClockIsFree();
outp(0x70, 0x00); // Schityvanie dannyh dlya secund
value = inp(0x71);
itoa((BCDToInteger(value)), str6, 10);
if (BCDToInteger(value) < 10)
{
        StaticLines("0", 6, 12);
        StaticLines(str6, 6, 13);
}
else {
        StaticLines(str6, 6, 12);
}

StaticLines("Date", 6, 18);
WaitClockIsFree();
outp(0x70, 0x07); // Schityvanie dannyh o dne
value = inp(0x71);
itoa((BCDToInteger(value)), str6, 10);
if (BCDToInteger(value) < 10)
{
        StaticLines("0", 6, 24);
        StaticLines(str6, 6, 25);
}
else {
        StaticLines(str6, 6, 24);
}
StaticLines(".", 6, 26);

WaitClockIsFree();
outp(0x70, 0x08); // Schityvanie dannyh o mesyace
value = inp(0x71);
itoa((BCDToInteger(value)), str6, 10);
if (BCDToInteger(value) < 10)
{
        StaticLines("0", 6, 27);
        StaticLines(str6, 6, 28);
}
else {
        StaticLines(str6, 6, 27);
}
StaticLines(".", 6, 29);

WaitClockIsFree();
```

```
        outp(0x70, 0x09); // Schityvanie dannyh o gode
        value = inp(0x71);
        itoa((BCDToInteger(value)), str6, 10);
        if (BCDToInteger(value) < 10)
        {
               StaticLines(str6, 6, 33);
               StaticLines("200", 6, 30);
        }
        else {
               StaticLines(str6, 6, 32);
               StaticLines("20", 6, 30);
        }

        WaitClockIsFree();
        outp(0x70, 0x06); // Schityvanie dannyh o dne nedeli
        value = inp(0x71);

        switch(BCDToInteger(value))
        {
               case 1:StaticLines("Sunday", 6, 36);  break;
               case 2:StaticLines("Monday", 6, 36);  break;
               case 3:StaticLines("Tuesday", 6, 36);  break;
               case 4:StaticLines("Wednesday", 6, 36);   break;
               case 5:StaticLines("Thursday", 6, 36);  break;
               case 6:StaticLines("Friday", 6, 36);   break;
               case 7:StaticLines("Saturday", 6, 36);   break;
               default:StaticLines("Undefined day of week", 6, 36);  break;
        }

}

void SetTime()
{
        int hours = 25, minutes = 60, seconds = 60, weekDay = 8, monthDay = 32, month = 13,
year = 21;
        int kolday[12] = { 31,28,31,30,31,30,31,31,30,31,30,31};
        char str[3];
        cleanmassive();
        disable();
        oldInt9 = getvect(9);    // Preryvanie klaviatury
        setvect(9, newInt9);
        enable();
        x = 20;
        gran = x;
        y = 7;
        StaticLines("Enter hours(0-23): ", 7, 0);

               while (hours < 0 || hours>23)  // Ustanovka chasov

               {

                      if (enter == 1)
                      {
                             hours = atoi(massive);
                             enter = 0;
                      }
               }

        cleanmassive();
        StaticLines("Enter minutes(0-59): ", 8, 0);
```

```
x = 20;
gran = x;
y++;
while (minutes < 0 || minutes>59)  // Ustanovka minut
{

	if (enter == 1)
	{
		minutes = atoi(massive);
		enter = 0;
	}
}
cleanmassive();
StaticLines("Enter seconds(0-59): ", 9, 0);
x = 20;
gran = x;
y++;
while (seconds < 0 || seconds>59) // Ustanovka secund
{

	if (enter == 1)
	{
		seconds = atoi(massive);
		enter = 0;
	}
}
cleanmassive();
StaticLines("Enter year(2000-2020): ", 10, 0);
x = 25;
gran = x;
y++;
while (year < 2000 || year > 2020)  // Ustanovka goda
{
	if (enter == 1)
	{
		year = atoi(massive);
		enter = 0;
	}
}
if (year % 4 != 0 || year % 100 == 0 && year % 400 != 0)
{}
else
	kolday[1] = 29;
year = year - 2000;
cleanmassive();
StaticLines("Enter month(1-12): ", 11, 0);
x = 20;
gran = x;
y++;
 while (month < 1 || month>12) // Ustanovka mesyaca
 {

	if (enter == 1)
	{
		month = atoi(massive);
		enter = 0;
	}
 }
cleanmassive();
itoa(kolday[month - 1], str, 10);
```

```
StaticLines("Enter day of month 1- ", 12, 0);
StaticLines(str, 12, 23);
x = 30;
gran = x;
y++;
while (monthDay < 1 || monthDay>kolday[month-1])  // Ustanovka dn'ya
{

	if (enter == 1)
	{
		monthDay = atoi(massive);
		enter = 0;
	}
}
cleanmassive();
StaticLines("Enter week day number(1-7): ", 13, 0);
x = 30;
gran = x;
y++;
while (weekDay < 1 || weekDay>7) // Ustanovka dn'ya nedely
{
	if (enter == 1)
	{
		weekDay = atoi(massive);
		enter = 0;
	}
}
cleanmassive();

setvect(9, oldInt9);

LockClockUpdate();  // Zapret na obnovlenie

outp(0x70, 0x04);
outp(0x71, IntToBCD(hours));

outp(0x70, 0x02);
outp(0x71, IntToBCD(minutes));

outp(0x70, 0x00);
outp(0x71, IntToBCD(seconds));

outp(0x70, 0x06);
outp(0x71, IntToBCD(weekDay));

outp(0x70, 0x07);
outp(0x71, IntToBCD(monthDay));

outp(0x70, 0x08);
outp(0x71, IntToBCD(month));

outp(0x70, 0x09);
outp(0x71, IntToBCD(year));

UnlockClockUpdate(); // Razreshenye na obnovlenie

StaticLines("Input  to continue", 14, 0);
getch();
CleanScreen(6, 720, 0);
inittimerstop();  // stop timer
```

```
}
void redakttime()
{
	int hours = 0, weekday = 8, monthday = 32, month = 13, year = 21;
	int kolday[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };
	unsigned char value;
	hours = 0;

	WaitClockIsFree();
	outp(0x70, 0x07); // day
	value = inp(0x71);
	monthday = BCDToInteger(value);

	WaitClockIsFree();
	outp(0x70, 0x08); // month
	value = inp(0x71);
	month = BCDToInteger(value);

	WaitClockIsFree();
	outp(0x70, 0x09); // year
	value = inp(0x71);
	year = BCDToInteger(value);
	year = year + 2000;
	if (year % 4 != 0 || year % 100 == 0 && year % 400 != 0)
	{
	}
	else
		kolday[1] = 29;

	WaitClockIsFree();
	outp(0x70, 0x06); // week day
	value = inp(0x71);
	weekday = BCDToInteger(value);

	weekday++;
	if (weekday > 7)
		weekday = 1;
	monthday++;
	if (monthday > kolday[month - 1]) {
		monthday = 1;
		month++;
		if (month > 12) {
			month = 1;
			year++;
		}
	}
	year = year - 2000;

	LockClockUpdate();

	outp(0x70, 0x04);
	outp(0x71, IntToBCD(hours));

	outp(0x70, 0x06);
	outp(0x71, IntToBCD(weekday));

	outp(0x70, 0x07);
	outp(0x71, IntToBCD(monthday));
```

```
        outp(0x70, 0x08);
        outp(0x71, IntToBCD(month));

        outp(0x70, 0x09);
        outp(0x71, IntToBCD(year));

        UnlockClockUpdate();

}
void LockClockUpdate()
{
        unsigned char value;
        WaitClockIsFree();  // Proverka chasov
        outp(0x70,0x0B);
        value = inp(0x71);  //  ustanovka 1 v 7 bite
        value|=0x80;         // Zapret obnovleniya chasov
        outp(0x70, 0x0B);
        outp(0x71, value);
}

void UnlockClockUpdate()
{
        unsigned char value;
        WaitClockIsFree();
        outp(0x70,0x0B);
        value = inp(0x71);   // ustanovka 0 v 7 bite
        value-=0x80;        // razreshenye na obnovleniya chasov
        outp(0x70, 0x0B);
        outp(0x71, value);
}

void interrupt far NewInt70Handler(void)
{
        msCounter++;

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

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

void CustomDelay()
{
        int delayPeriod=0;
        unsigned char value;
        StaticLines("Delay in ms (input between 1000 and 10000):  ", 7, 0);
        cleanmassive();
        disable();
        oldInt9 = getvect(9);    // Preryvanie klaviatury
        setvect(9, newInt9);
        enable();
        x = 45;
        gran = x;
        y = 7;
                while (delayPeriod < 1000 || delayPeriod > 10000)
                {

                        if (enter == 1)
                        {
```

```
                        delayPeriod = atoi(massive);
                        enter = 0;
                    }
                }
        setvect(9, oldInt9);
        cleanmassive();
        setvect(0x1c, oldTimerHandler);
        disable();
        oldInt70h = getvect(0x70);
        setvect(0x70, NewInt70Handler);
        enable();

        value = inp(0xA1);
        outp(0xA1, value & 0xFE);  // Chasy zanyaty

        outp(0x70, 0x0B);
        value = inp(0x0B);
        outp(0x70, 0x0B);
        outp(0x71, value & 0x40);  // Vyzov periodicheskogo preryvaniya

        msCounter = 0;
        while (msCounter != delayPeriod);
        StaticLines("End delay", 8, 0);

        setvect(0x70, oldInt70h);
        WaitClockIsFree();
        setvect(0x1c,TimerHandler);

        StaticLines("Input  to continue", 10, 0);
        getch();
        CleanScreen(6, 720, 0);
        inittimerstop();
}



int BCDToInteger (int bcd)
{
        return bcd % 16 + bcd / 16 * 10;  // perevod v int
}

unsigned char IntToBCD (int value)
{
        return (unsigned char)((value/10)<<4)|(value%10); // perevod v BCD
}
void Vprint(char *str)  // vyvod
{
        int i;
        char far* start = (char far*)0xb8000000;
        char far *v;
        for (i = 0; str[i] != '\0'; i++)
        {
                v = start + y * 160 + x * 2;
                x++;
                *v = str[i];
                v++;
                *v = 0x0F;
        }

}
```

```
void StaticLines(char *str, int line, int column)  // vyvod
{
	int i;
	char far* start = (char far*)0xb8000000;
	char far *v;
	for (i = 0; str[i] != '\0'; i++)
	{

		v = start + line * 160 + column * 2;
		column++;
		*v = str[i];
		v++;
		*v = 7;
	}

}
void CleanScreen(int line, int counts, int column)
{
	int i;
	char far* start = (char far*)0xb8000000;
	char far *v;
	for (i = 0; i<counts; i++)
	{
		v = start + line * 160 + column * 2;
		column++;
		*v = ' ';
		v++;
	}
}


void cleanmassive()
{
	int i;
	for (i = 0; i < 20; i++) {
		massive[i] = '\0';
	}
	nomer = 0;
}
```

5. Копии экрана выполнения программы

После запуска программы на экран выводится головная информация, значения регистров и меню программы (рисунок 5.1). При выборе пункта «показать время» на экран над меню выводится текущее время, дата и день недели (рисунок 5.2). При выборе «изменить время» пользователь может ввести данные о новых дате и времени (рисунок 5.3). При выборе «задержка» пользователь может задать задержку (рисунок 5.4). При нажатии на «Esc» происходит выход из программы.

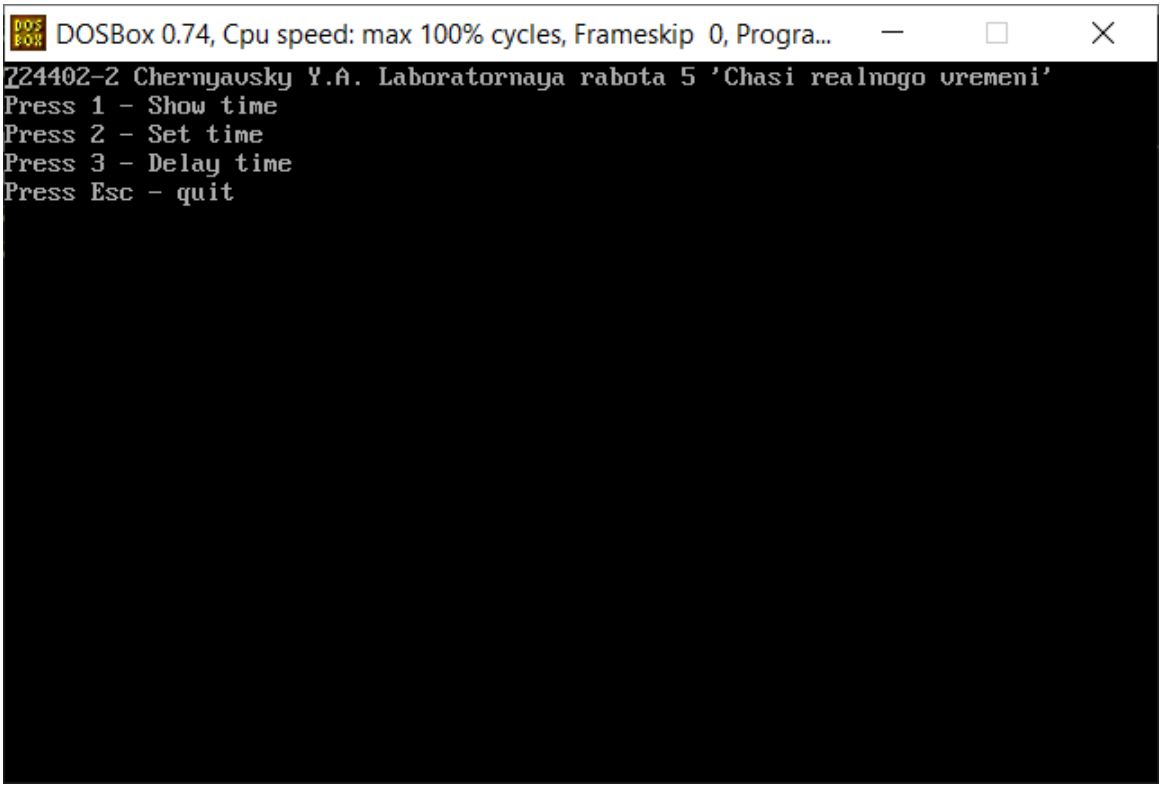

Рисунок 5.1 – Меню программы

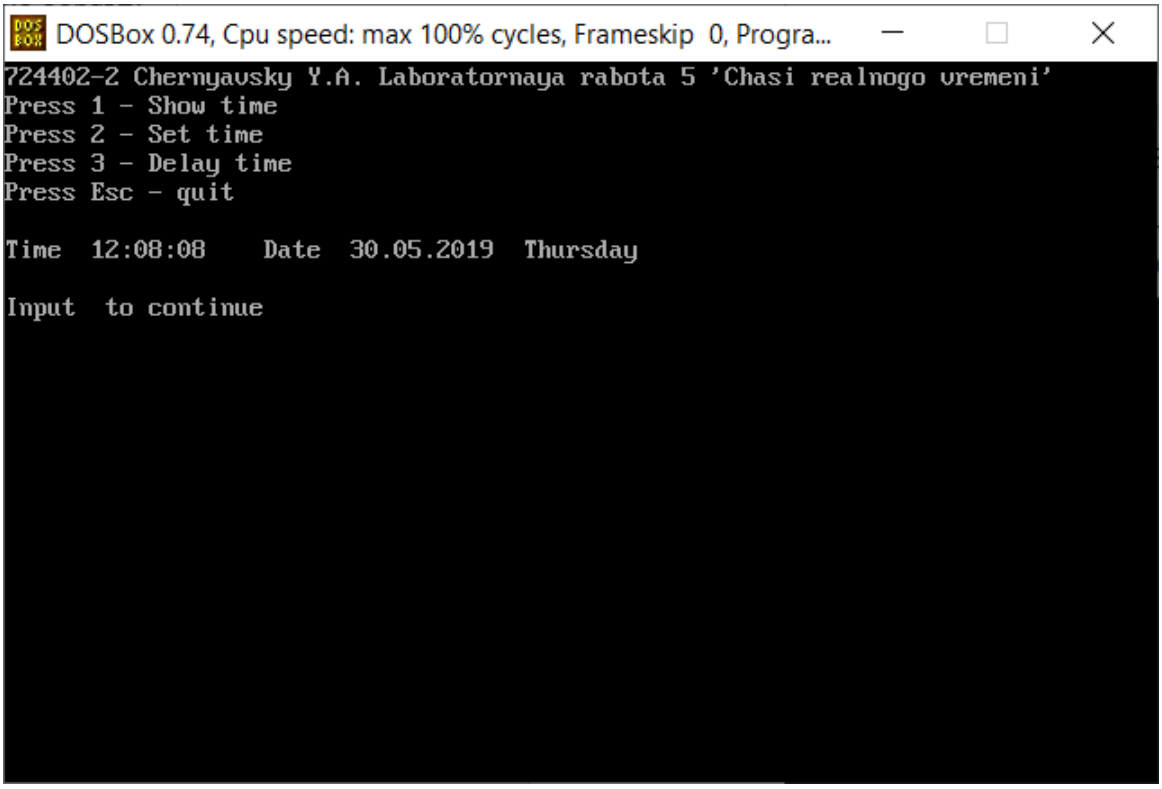

Рисунок 5.2 – Вывод текущих даты и времени

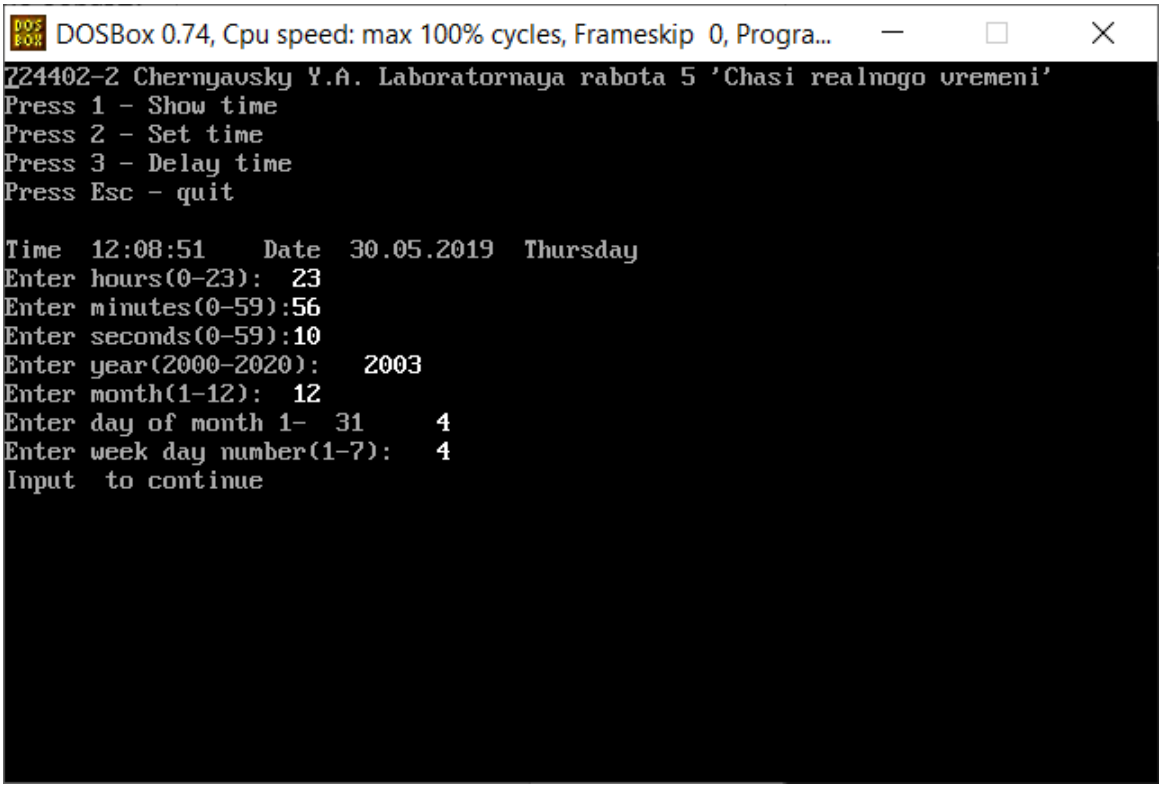

Рисунок 5.3 – Измнение времени

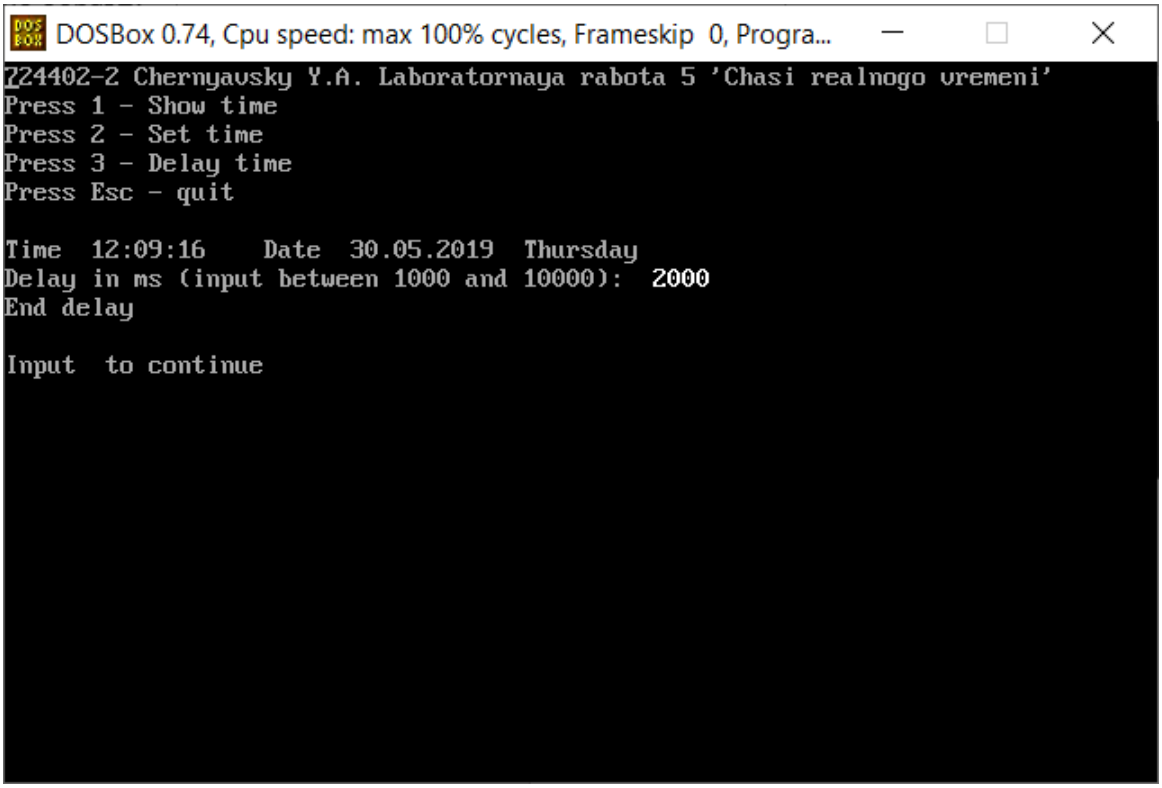

Рисунок 5.4 – Задержка времени

6. Вывод

В результате выполнения работы были получены навыки программирования и изучены возможности использования часов реального времени, написаны свои собственные обработчики прерываний, выведено содержимое регистров состояния RTC A, B, C, D на экран с помощью работы с видеобуфером, а также реализованы функции считывания, установки времени и установки задержки.