Министерство образования Республики Беларусь

Учреждение образования Белорусский государственный университет

информатики и радиоэлектроники

# Кафедра ЭВМ

### Отчет по лабораторной работе № 5

"Программирование часов реального времени"

Выполнил:

студент группы 950502

Киреев Ю. В.

Проверил:

Одинец Д. Н.

#### Минск 2021

1. **Задание**

Задание состоит из двух частей. Первая часть общая для всех. Вторая часть по вариантам. Варианты выдаются преподавателем после того, как студенты разделятся на бригады.

**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.

*Условия выполнения данного варианта:*

1.1. Задержка должна вводится с клавиатуры в миллисекундах в удобной для пользователя форме.

1.2. После окончания периода задержки программа должна сообщить об этом в любой форме. При этом зависание компьютера не допускается.

2. Используя аппаратное прерывания часов реального времени и режим будильника реализовать функции программируемого будильника.

2.1. Время будильника вводится с клавиатуры в удобной для пользователя форме.

2.2. При срабатывании 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.

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 };

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

char\* months[] =

{

"JANUARY",

"FEBRUARY",

"MARCH",

"APRIL",

"MAY",

"JUNE",

"JULY",

"AUGUST",

"SEPTEMBER",

"OCTOBER",

"NOVEMBER",

"DECEMBER"

};

void menu();

void setTime();

void showTime();

void setAlarm();

void enterTime();

void resetAlarm();

int convertToBCD(int);

void delay(unsigned int);

int convertToDecimal(int);

int main()

{

while (1)

{

menu();

switch (getch())

{

case '1':

system("cls");

showTime();

break;

case '2':

system("cls");

int delayms;

printf("INPUT DELAY (MS): ");

scanf("%d", &delayms);

delay(delayms);

break;

case '3':

system("cls");

setAlarm();

break;

case '4':

system("cls");

setTime();

break;

case '5':

return 0;

default:

system("cls");

break;

}

}

}

void menu()

{

puts("=====================");

puts("1. SHOW TIME");

puts("2. SET DELAY");

puts("3. SET ALARM");

puts("4. SET TIME");

puts("5. EXIT");

puts("=====================");

}

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

printf("%2d:%2d:%2d\n", decimalData[2], decimalData[1], decimalData[0]); // час, минута, секунда

printf("%2d %s 20%2d\n", decimalData[3], months[decimalData[4] - 1], decimalData[5]); // день, месяц, год

}

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 delay(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);

puts("DELAY IS OVER");

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

puts("ALARM IS ON");

}

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 enterTime()

{

int enter;

do

{

rewind(stdin);

printf("ENTER YEAR: ");

scanf("%i", &enter);

} while ((enter > 2100 || enter < 0));

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 > 365 || 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 MINUTES: ");

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);

}

int convertToDecimal(int BCD)

{

return ((BCD / 16 \* 10) + (BCD % 16));

}

int convertToBCD(int decimal)

{

return ((decimal / 10 \* 16) + (decimal % 10));

}

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

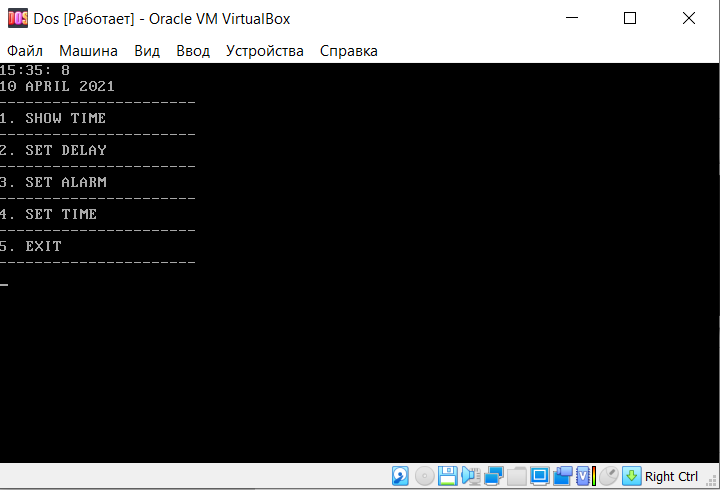
{

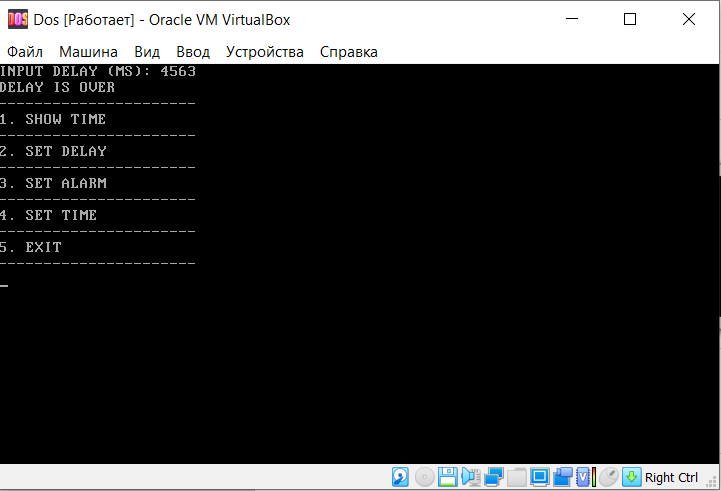
puts("ALARM! ALARM! ALARM!");

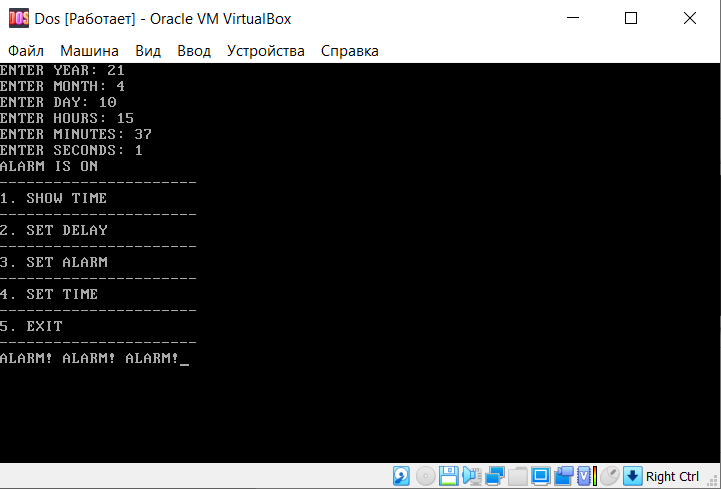
lastAlarm();

resetAlarm();

}







**Вывод:** в ходе лабораторной работы было выполнено программирование часов реального времени.