Министерство образования Республики Беларусь

Учреждение образования Белорусский государственный университет информатики и радиоэлектроники

Кафедра ЭВМ

Отчёт по лабораторной работе №1

“Программирование контроллера прерываний”

Проверил: Выполнил:

к.т.н., доцент студент гр.250503

Одинец Дмитрий Николаевич Патюпин М.С.

Минск 2024

**Задача**

Целью данной работы является разработка программного модулья реализации процедуры передачи (приёма) байта информации через последовательный интерфейс.

Программа должна демонстрировать программное взаимодействие с последовательным интерфейсом с использованием следующих механизмов:

1. Прямое взаимодействие с портами ввода-вывода (wirte, read)

2. Использование BIOS прерывания 14h

3. Работа с COM-портом через регистры как с устройствами ввода-вывода.

**Алгоритм**

Программа состоит из нескольких подпрограмм (частей программы), представляющих собой некоторые функции. К ним относятся функции:

* Инициализация порта
* Запись байта информации в порт
* Чтение байта информации из порта
* Вывод результата на экран

**Листинг программы, взаимодействующей с портами ввода-вывода**

#include <windows.h>

#include <iostream>

using namespace std;

HANDLE COM\_Port\_1;

LPCTSTR Port\_Name\_1 = L"COM1";

HANDLE COM\_Port\_2;

LPCTSTR Port\_Name\_2 = L"COM2";

void Read\_from\_COM()

{

DWORD Size;

char Received\_Char;

ReadFile(COM\_Port\_2, &Received\_Char, 1, &Size, 0);

if (Size > 0)

{

cout << Received\_Char;

}

}

int main()

{

COM\_Port\_1 = ::CreateFile(Port\_Name\_1, GENERIC\_WRITE, 0, 0, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, 0);

COM\_Port\_2 = ::CreateFile(Port\_Name\_2, GENERIC\_READ, 0, 0, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, 0);

if (COM\_Port\_1 == INVALID\_HANDLE\_VALUE)

{

if (GetLastError() == ERROR\_FILE\_NOT\_FOUND)

{

cout << "COM-port does not exist!\n";

}

cout << "Some other error.\n";

}

DCB Serial\_Params = { 0 };

Serial\_Params.DCBlength = sizeof(Serial\_Params);

if (!GetCommState(COM\_Port\_1, &Serial\_Params))

{

cout << "Getting state error.\n";

}

Serial\_Params.BaudRate = CBR\_9600;

Serial\_Params.ByteSize = 8;

Serial\_Params.StopBits = ONESTOPBIT;

Serial\_Params.Parity = NOPARITY;

if (!SetCommState(COM\_Port\_2, &Serial\_Params))

{

cout << "Error setting serial port state.\n";

}

char data = 'H';

DWORD Size = sizeof(data);

DWORD Bytes\_Written;

BOOL Ret = WriteFile(COM\_Port\_1, &data, Size, &Bytes\_Written, NULL);

cout << Size << " Bytes in string. " << Bytes\_Written << " Bytes sended. " << endl;

Read\_from\_COM();

return 0;

}

**Листинг программы, использующей BIOS прерывание 14h**

.model small

.stack 100h

.data

Error\_Write db "Write error!",0Dh,0Ah,'$' ; Сообщение об ошибке записи

Error\_Read db "Read error!",0Dh,0Ah,'$' ; Сообщение об ошибке чтения

Information db "Byte sent: $" ; Сообщение о переданном байте

.code

jmp start ; Переход к началу программы

Init\_COM1 proc

xor ax,ax ; Обнуление регистра AX

mov al,10100011b ; Загрузка значения AL для инициализации COM-порта

mov dx,0 ; Установка значения DX в 0

int 14h ; Вызов прерывания 14h для инициализации COM-порта

ret ; Возврат из процедуры

Init\_COM1 endp

IsWrite\_COM1 proc

mov al,'A' ; Загрузка значения AL 'A'

mov ah,1 ; Установка значения AH в 1 для проверки записи

mov dx,0 ; Установка значения DX в 0

int 14h ; Вызов прерывания 14h для проверки возможности записи

test al,80h ; Проверка, установлен ли бит записи

jnz NoWrite ; Переход к NoWrite, если запись невозможна

ret ; Возврат из процедуры

IsWrite\_COM1 endp

NoWrite proc

mov ah,9 ; Установка значения AH в 9 для вывода строки

mov dx,offset Error\_Write ; Загрузка адреса сообщения об ошибке записи

add dx,2 ; Коррекция DX для пропуска байта длины строки

int 21h ; Вызов прерывания 21h для вывода сообщения об ошибке

ret ; Возврат из процедуры

NoWrite endp

IsRead\_COM2 proc

mov ah,2 ; Установка значения AH в 2 для проверки чтения

mov dx,1 ; Установка значения DX в 1 для COM2

int 14h ; Вызов прерывания 14h для проверки возможности чтения

test al,80h ; Проверка, установлен ли бит чтения

jnz NoRead ; Переход к NoRead, если чтение невозможно

ret ; Возврат из процедуры

IsRead\_COM2 endp

NoRead proc

mov ah,9 ; Установка значения AH в 9 для вывода строки

mov dx,offset Error\_Read ; Загрузка адреса сообщения об ошибке чтения

add dx,2 ; Коррекция DX для пропуска байта длины строки

int 21h ; Вызов прерывания 21h для вывода сообщения об ошибке

ret ; Возврат из процедуры

NoRead endp

Output proc

mov ah,02h ; Установка значения AH в 02h для вывода символа

mov dl,al ; Копирование значения из AL в DL

int 21h ; Вызов прерывания 21h для вывода символа

ret ; Возврат из процедуры

Output endp

Exit proc

mov ax,4C00h ; Установка значения AX в 4C00h для завершения программы

int 21h ; Вызов прерывания 21h для завершения программы

ret ; Возврат из процедуры

Exit endp

start:

call Init\_COM1 ; Вызов процедуры инициализации COM1-порта

call IsWrite\_COM1 ; Вызов процедуры проверки возможности записи

mov al,'e' ; Установка значения AL в 'e'

call IsRead\_COM2 ; Вызов процедуры проверки возможности чтения

push ax ; Сохранение значения AX в стеке

mov ah,9 ; Установка значения AH в 9 для вывода строки

mov dx,offset Information ; Загрузка адреса информационного сообщения

add dx,2 ; Коррекция DX для пропуска байта длины строки

int 21h ; Вызов прерывания 21h для вывода информационного сообщения

pop ax ; Извлечение значения AX из стека

call Output ; Вызов процедуры вывода символа

call Exit ; Вызов процедуры завершения программы

end start

**Листинг программы, работающей с COM-портами через регистры как с устройствами ввода-вывода**

.model small

.stack 100h

.data

Error\_Write db "Write error!",0Dh,0Ah,'$' *; Сообщение об ошибке записи*

Error\_Read db "Read error!",0Dh,0Ah,'$' *; Сообщение об ошибке чтения*

Information db "Byte sent: $" *; Сообщение об отправленном байте*

Data\_Byte db 'A' *; Байт для отправки*

Data\_Byte2 db ? *; Прочитанный байт*

.code

Init\_COM1 proc

mov al,80h *; Устанавливаем бит 7 в регистре AL, чтобы активировать биты 1 и 2 для порта 3FBh*

mov dx,3FBh ; *Порт 3FBh - регистр управления COM-портом*

out dx,al ; *Записываем значение AL в порт DX*

mov dx,3F8h ; *Порт 3F8h - регистр данных COM-порта*

mov al,00h ; *Очищаем регистр данных*

out dx,al ; *Записываем значение AL в порт DX*

mov al,0Ch ; *Настраиваем биты 3 и 4 для порта 3F9h*

mov dx,3F9h ; *Порт 3F9h - регистр управления COM-портом*

out dx,al ; *Записываем значение AL в порт DX*

mov dx,3FCh ; *Порт 3FCh - регистр линий для COM-порта*

mov al,00001011b ; *Настраиваем биты 0, 1 и 3*

out dx,al ; *Записываем значение AL в порт DX*

mov dx,3F9h ; *Порт 3F9h - регистр управления COM-портом*

mov al,0 ; *Очищаем регистр управления*

out dx,al ; *Записываем значение AL в порт DX*

ret

Init\_COM1 endp

IsWrite\_COM1 proc

xor al,al ; *Сбрасываем AL в 0*

mov dx,3FDh ; *Порт 3FDh - регистр статуса COM-порта*

in al,dx ; *Читаем значение из порта DX в AL*

test al,10h ; *Проверяем, установлен ли бит 4*

jnz NoWrite ; *Если бит 4 установлен, переходим к процедуре NoWrite*

ret ; *Возвращаемся*

IsWrite\_COM1 endp

NoWrite proc

mov ah,9 ; Устанавливаем функцию вывода строки

mov dx,offset Error\_Write ; Загружаем адрес сообщения об ошибке записи

int 21h ; Вызываем прерывание 21h для вывода сообщения

ret ; Возвращаемся

NoWrite endp

IsRead\_COM2 proc

xor al,al ; Сбрасываем AL в 0

mov dx,3FDh ; Порт 3FDh - регистр статуса COM-порта

in al,dx ; Читаем значение из порта DX в AL

test al,10b ; Проверяем, установлен ли бит 4

jnz NoRead ; Если бит 4 установлен, переходим к процедуре NoRead

ret ; Возвращаемся

IsRead\_COM2 endp

NoRead proc

mov ah,9 ; Устанавливаем функцию вывода строки

mov dx,offset Error\_Read ; Загружаем адрес сообщения об ошибке чтения

int 21h ; Вызываем прерывание 21h для вывода сообщения

ret ; Возвращаемся

NoRead endp

Send\_Byte proc

mov dx,3F8h ; Порт 3F8h - регистр данных COM-порта

mov al,Data\_Byte ; Загружаем байт для отправки в AL

out dx,al ; Записываем значение AL в порт DX

ret ; Возвращаемся

Send\_Byte endp

Read\_Byte proc

mov dx,3F8h ; Порт 3F8h; Порт 3F8h - регистр данных COM-порта

in al,dx ; Читаем значение из порта DX в AL

mov Data\_Byte2,al ; Сохраняем прочитанный байт в переменной Data\_Byte2

ret ; Возвращаемся

Read\_Byte endp

Exit proc

mov ax,4C00h ; Загружаем значение для завершения программы

int 21h ; Вызываем прерывание 21h для завершения программы

ret ; Возвращаемся

Exit endp

start:

mov ax,@data ; Загружаем адрес сегмента данных в регистр AX

mov ds,ax ; Загружаем значение регистра AX в регистр DS

call Init\_COM1 ; Вызываем процедуру инициализации COM-порта

call IsWrite\_COM1 ; Проверяем, можно ли записывать в COM-порт

call Send\_Byte ; Отправляем байт в COM-порт

mov al,2 ; Устанавливаем значение AL в 2

call IsRead\_COM2 ; Проверяем, можно ли читать из COM-порта

call Read\_Byte ; Читаем байт из COM-порта

mov dx,offset Information ; Загружаем адрес строки с информацией

mov ah,9 ; Устанавливаем функцию вывода строки

int 21h ; Вызываем прерывание 21h для вывода сообщения

mov ah,02h ; Устанавливаем функцию вывода символа

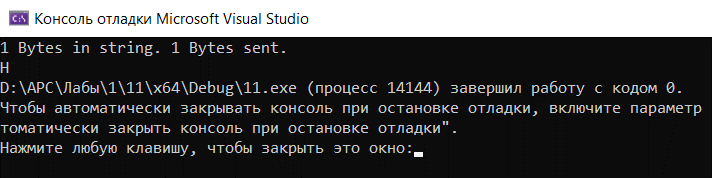
mov dl,Data\_Byte2 ; Загружаем прочитанный байт в DL

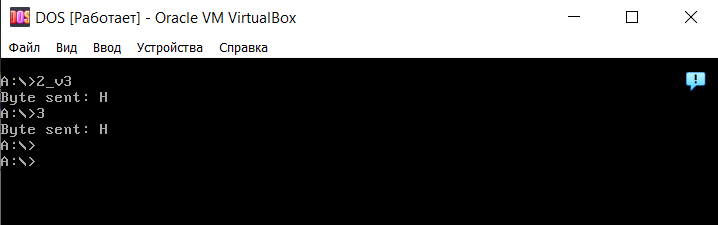
int 21h ; Вызываем прерывание 21h для вывода символа

call Exit ; Вызываем процедуру завершения программы

end start

**Тест**





**Заключение**

В ходе лабораторной удалось передать 1 байт информации через последовательный порт с использованием различных механизмов.

Выполнена с использованием emu8086 для написания программы на языке ассемблера, для эмуляции MS-Dos использовалась программное обеспечение “Oracle VM VirtualBox” на хосте Windows 10.