*Федеральное государственное бюджетное образовательное учреждение* *высшего образования*

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана***  ***(национальный исследовательский университет)»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ \_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Отчет**

**по лабораторной работе № \_5\_\_\_\_**

**Дисциплина: \_**Машинно-зависимые языки и основы компиляции**\_\_\_\_**

**Название лабораторной работы: \_**Программирование с использованием\_

\_\_разноязыковых модулей\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент гр. **\_**ИУ6-42б**\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_**И.С. Марчук**\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Москва, 2021

**Задание:**

Разработать программу, состоящую из трех модулей:

1) основной модуль на выбранном языке программирования общего назначения должен содержать диалоговый ввод необходимых данных, вызов функции или процедуры на ассемблере и вывод результатов;

2) второй модуль - функция или процедура на ассемблере, выполняющая заданную обработку и вызывающая для печати диагностических сообщений процедуру на выбранном языке программирования общего назначения;

3) третий модуль - процедура на выбранном языке общего назначения, обязательно получающая некоторые параметры из вызывающего модуля.

**Вариант 16**. Дан текст не более 255 символов. Слова отделяются друг от друга пробелами.

Поменять местами пары слов с указанными номерами.

**Цель работы:**

Изучение конвенций о способах передачи управления и данных при вызове из программы, написанной на языке высокого уровня, подпрограмм, написанных на ассемблере.

**Схема алгоритма для решения поставленной задачи представлена на рисунках 1-4:**

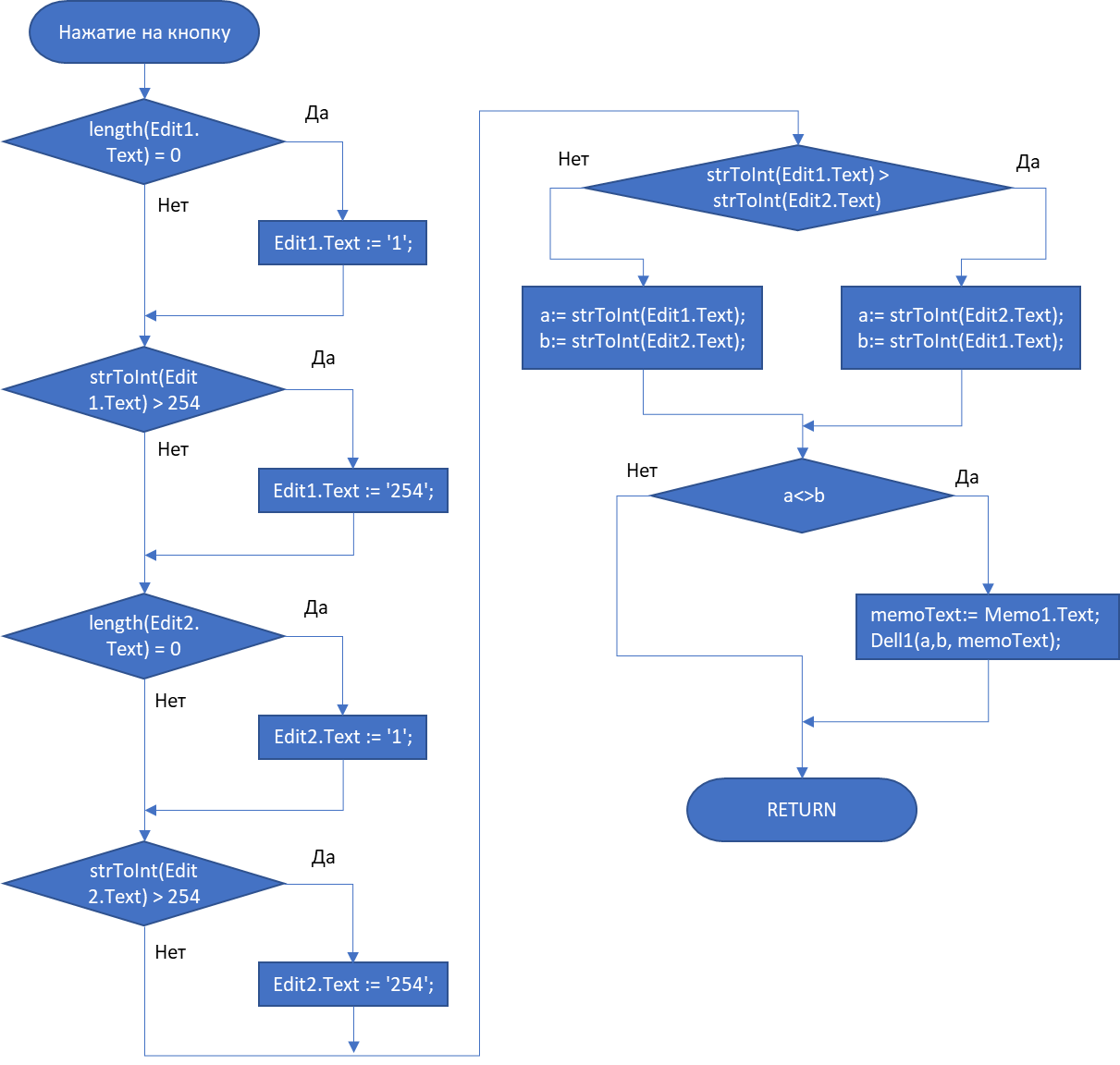


Рисунок 1 – Схема алгоритма основной программы на паскале

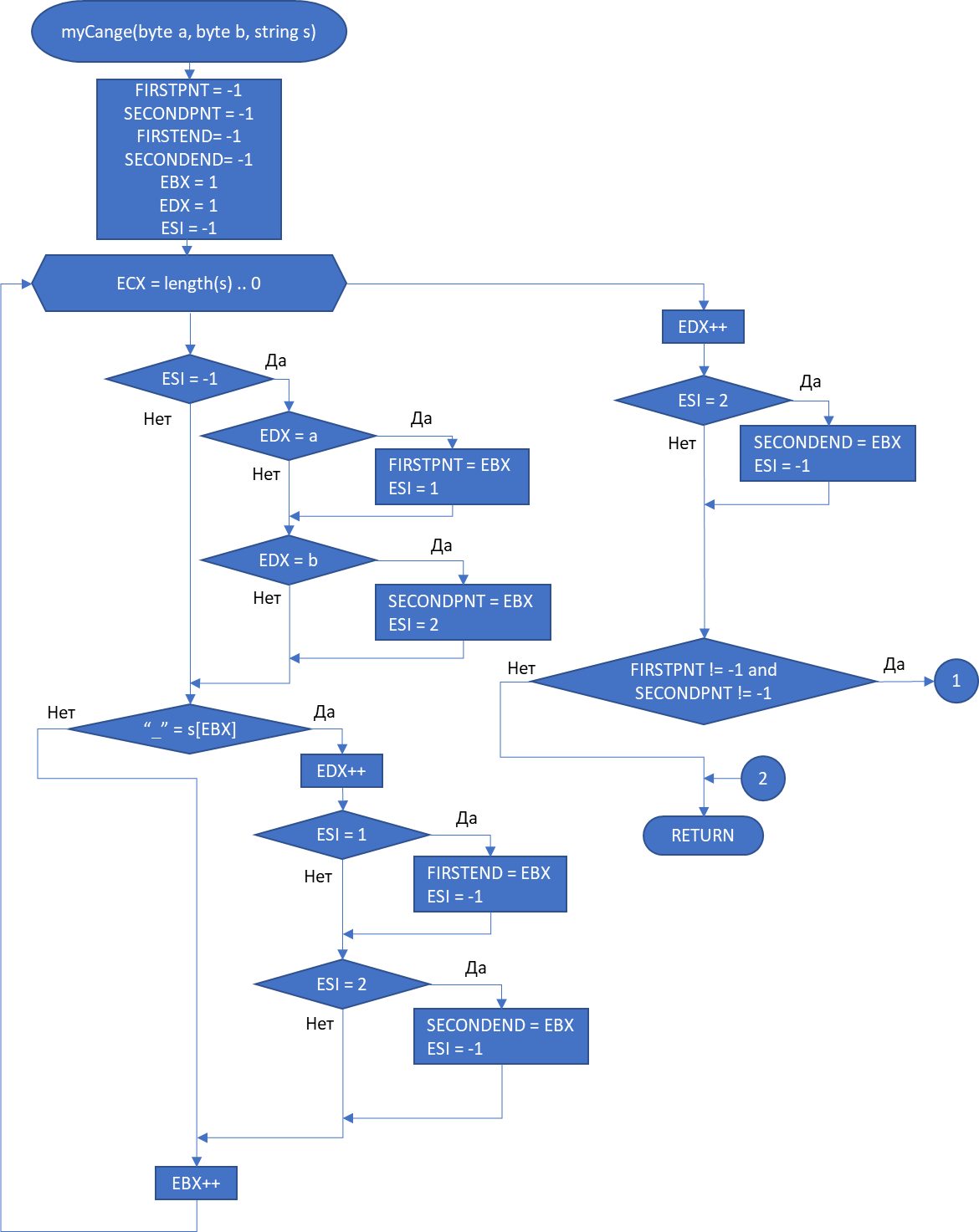


Рисунок 2 – Схема алгоритма подпрограммы на ассемблере(ч. 1)

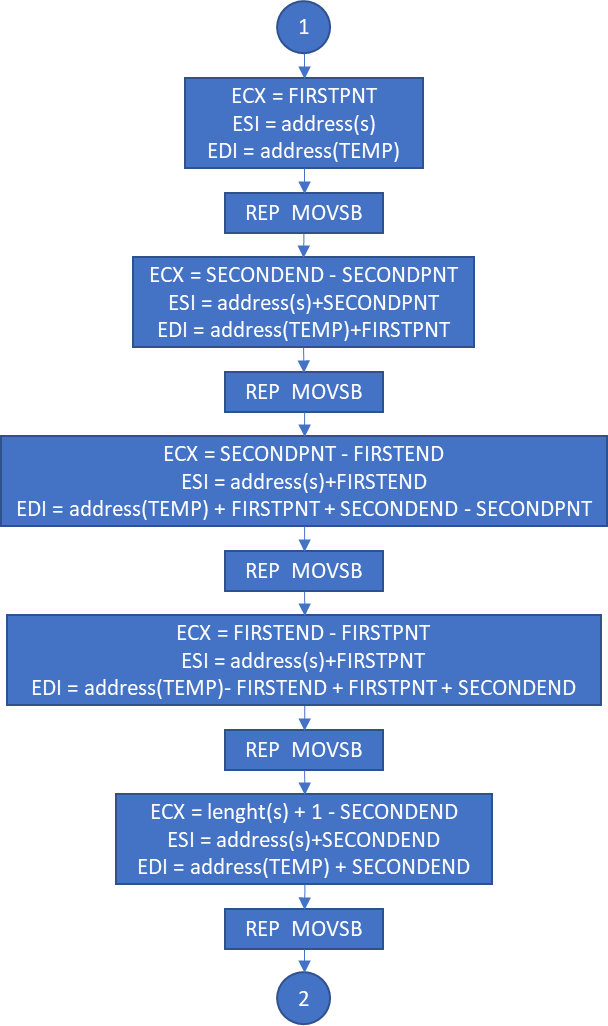


Рисунок 3 – Схема алгоритма подпрограммы на ассемблере(ч. 2)

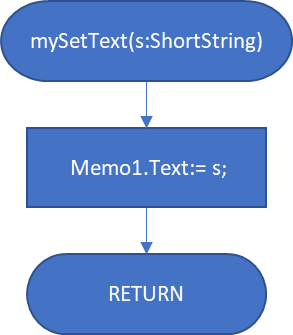


Рисунок 4 – Схема алгоритма подпрограммы на паскале вызываемой из подпрограммы на ассемблере

**Текст программы с комментариями:**

Модуль Lasarus с вызывающей процедурой срабатывающий при нажатии на форму, процедура в нем вызываемая через ассемблер, и объявление ассемблерной процедуры:

unit Unit1;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, ExtCtrls;

type

{ TForm1 }

TForm1 = class(TForm)

Button1: TButton;

Edit1: TEdit;

Edit2: TEdit;

Memo1: TMemo;

Panel1: TPanel;

Label1 : TLabel;

Label2 : TLabel;

Label3 : TLabel;

procedure Button1Click(Sender: TObject);

procedure Edit1KeyPress(Sender: TObject; var Key: char);

procedure Edit2KeyPress(Sender: TObject; var Key: char);

procedure FormCreate(Sender: TObject);

procedure Memo1Change(Sender: TObject);

procedure mySetText(s:ShortString);

private

public

end;

var

Form1: TForm1;

implementation

{$R \*.lfm}

{$L lab05assemb\lab05assemb.OBJ}

// ассемблерная процедура изменения текста

procedure myCange(a,b:byte; s:ShortString);pascal;external;

// pascal процедура назначения нового текста

procedure TForm1.mySetText(s:ShortString);

begin

Memo1.Text:= s;

end;

{ TForm1 }

procedure TForm1.FormCreate(Sender: TObject);

begin

end;

procedure TForm1.Memo1Change(Sender: TObject);

begin

if(length(Memo1.Text) > 255)then

begin

Memo1.Text := copy(Memo1.Text, 1, 255);

end;

end;

procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: char);

begin

if not (Key in ['0'..'9', #8]) then Key := #0;

end;

procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: char);

begin

if not (Key in ['0'..'9', #8]) then Key := #0;

end;

procedure TForm1.Button1Click(Sender: TObject);

var

memoText:ShortString;

a,b:shortint;

begin

// проверка на допустимые значения

if(length(Edit1.Text) = 0)then

begin

Edit1.Text := '1';

end;

if(strToInt(Edit1.Text) > 254)then

begin

Edit1.Text := '254';

end;

if(length(Edit2.Text) = 0)then

begin

Edit2.Text := '1';

end;

if(strToInt(Edit2.Text) > 254)then

begin

Edit2.Text := '254';

end;

if(strToInt(Edit1.Text) > strToInt(Edit2.Text))then

begin

a:= strToInt(Edit2.Text);

b:= strToInt(Edit1.Text);

end

else

begin

a:= strToInt(Edit1.Text);

b:= strToInt(Edit2.Text);

end;

// вызов подпрограммы

if(a<>b)then

begin

memoText:= Memo1.Text;

myCange(a,b, memoText);

end;

end;

end.

**Код ассемблера содержащий в себе процедуру вызываемую через pascal:**

.586

.MODEL flat, pascal

.CONST

MSEPARATE DWORD " ",0

.DATA

FIRSTPNT DWORD -1

SECONDPNT DWORD -1

FIRSTEND DWORD -1

SECONDEND DWORD -1

.DATA?

TEMP DB 255 DUP (?)

.CODE

PUBLIC \_myCange

EXTERNDEF mySetText:NEAR

\_myCange PROC

PUSH EBP

MOV EBP, ESP

ADD ESP,256 ;выделение памяти под копию строки

; сохранение регистров

PUSH EBX

PUSH ESI

PUSH EDI

; копируем строку в стек

mov ESI,[EBP+8] ; адрес исходной строки

lea EDI,[EBP-256]; адрес копии исходной строки

xor ECX,ECX

mov CL, [ESI] ; длина исходной строки

inc ECX ; добавим адрес начала исходной строки

rep movsb ; копирование исходной строки

MOV EBX, [EBP-256]; адрес исходной строки

XOR ECX, ECX

MOV CL, [EBX]; начинаем цикл по длинне всей строки, в которой находим позиции необходимых слов

MOV EBX, 1; счетчик символов

MOV EDX, 1; счетчик количества слов

MOV ESI, -1; флаг обработки слова

search\_loop:

CMP ESI, -1; проверяем не идет ли уже на этом слове проверка

JNZ end\_sec\_if

CMP EDX, [EBP+16] ;если мы находимся на слове находящемся под первым номером

JNZ end\_first\_if

MOV FIRSTPNT, EBX; копируем найденный адрес

MOV ESI, 1; флаг

end\_first\_if:

CMP EDX, [EBP+12] ;если мы находимся на слове находящемся под вторым номером

JNZ end\_sec\_if

MOV SECONDPNT, EBX; копируем найденный адрес

MOV ESI, 2; флаг

end\_sec\_if:

MOV EDI, [EBP-256] ; получаем базовый адрес строки

INC EDI

XOR EAX, EAX

MOV AL, [EDI+EBX-1]; получаем текущий элемент

CMP MSEPARATE, EAX; если находим пробел увеличиваем счетчик слов

JNZ end\_numb\_if

INC EDX

CMP ESI, 1; находим конец первой строки

JNZ end\_f\_end\_if

MOV FIRSTEND, EBX; копируем найденный адрес

MOV ESI, -1; флаг

end\_f\_end\_if:

CMP ESI, 2; находим конец второй строки

JNZ end\_s\_end\_if

MOV SECONDEND, EBX; копируем найденный адрес

MOV ESI, -1; флаг

end\_s\_end\_if:

end\_numb\_if:

INC EBX

LOOP search\_loop

INC EDX ; после цикла последний раз увеличиваем счетчик слов

CMP ESI, 2; находим конец второй строки

JNZ end\_s\_d\_end\_if

MOV SECONDEND, EBX; копируем найденный адрес

MOV ESI, -1; флаг

end\_s\_d\_end\_if:

MOV EAX, -1 ; после выполнения цикла имеем позиции слов или -1 в памяти

CMP FIRSTPNT, EAX ; в случае -1 переходим в конец программы без переставления слов

JE end\_p

CMP SECONDPNT, EAX

JE end\_p

; меняем слова местами

; копируем в новую переменную строку до начала первого слова

MOV ECX, FIRSTPNT

MOV ESI, [EBP-256] ; получаем базовый адрес изначальной строки

LEA EDI, TEMP; получаем базовый адрес новой строки

REP MOVSB

; копируем второе слово

MOV ECX, SECONDEND ; количество букв во втором слове

SUB ECX, SECONDPNT

PUSH ECX ; количество букв еще пригодится

; счетчик позиции в старой строке

MOV ESI, [EBP-256] ; база

ADD ESI, SECONDPNT ; индекс

; счетчик позиции в новой строке

LEA EDI, TEMP ; база

ADD EDI, FIRSTPNT ; индекс

REP MOVSB

; копируем часть цепочки от конца первого слова до начала второго слова

; счетчик позиции в старой строке

MOV ESI, [EBP-256] ; база

ADD ESI, FIRSTEND ; индекс

; счетчик позиции в новой строке

LEA EDI, TEMP ; база

MOV EAX, FIRSTPNT ; индекс

POP ECX ; достаем количество букв второго слова

ADD EAX, ECX

PUSH EAX ; Сохраняем позицию в новой строке для дальнейшей обработки

ADD EDI, EAX

; количество символов

MOV ECX, SECONDPNT

SUB ECX, FIRSTEND

REP MOVSB

; копируем первое слово

; счетчик позиции в старой строке

MOV ESI, [EBP-256] ; база

ADD ESI, FIRSTPNT ; индекс

; счетчик позиции в новой строке

LEA EDI, TEMP ; база

POP EAX ; достаем предыдущую позицию и прибавляем к ней расстояние между словами

ADD EAX, SECONDPNT ; индекс

SUB EAX, FIRSTEND

ADD EDI, EAX

; количество символов

MOV ECX, FIRSTEND

SUB ECX, FIRSTPNT

REP MOVSB

; копируем от конца второго слова до конца цепочки

; счетчик позиции в старой строке

MOV ESI, [EBP-256] ; база

ADD ESI, SECONDEND ; индекс

; счетчик позиции в новой строке

LEA EDI, TEMP ; база

ADD EDI, SECONDEND ; индекс

MOV EBX, [EBP-256]; адрес исходной строки

XOR ECX, ECX

MOV CL, [EBX] ; количество символов в исходной строке

ADD CL, 1

SUB ECX, SECONDEND

REP MOVSB

; возвращаем сохраненные регистры

PUSH EBX

PUSH ESI

PUSH EDI

; через внешнюю процедуру меняем текст

LEA EAX, TEMP

PUSH EAX

CALL myCange

end\_p:

POP EBP

RET 16

\_myCange ENDP

END

**Работа программы с тестовыми данными представлена в таблице 1 и рисунке 5:**

|  |  |  |
| --- | --- | --- |
| Исходные данные | Ожидаемый результат | Полученный результат |
| 1 22 3 4 5 6 7 8 9  2  5 | 1 5 3 4 22 6 7 8 9 | 1 5 3 4 22 6 7 8 9  Press Enter to Exit |
| 1 22 3 4 5 6 7 8 9  1  5 | 5 22 3 4 1 6 7 8 9 | 5 22 3 4 1 6 7 8 9  Press Enter to Exit |
| it is my word hello program  4  5 | it is my hello word program | it is my hello word program  Press Enter to Exit |
| it is my word hello program  4  6 | it is my program hello word | it is my program hello word  Press Enter to Exit |

Таблица 1 – Отладка программы

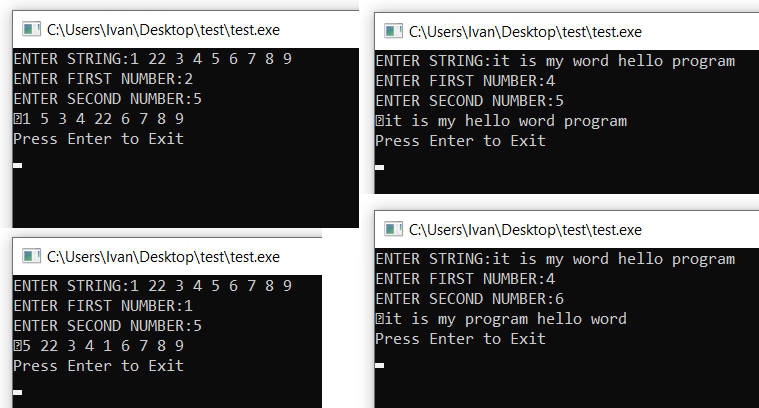


Рисунок 5 – Вывод ассемблерной процедуры скомпилированной в MASM32

На рисунке 6 представлена форма ввода значений и текста

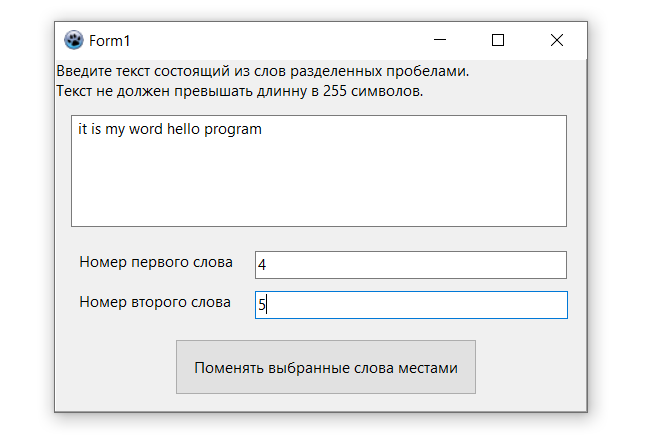


Рисунок 6 – Форма ввода

**Контрольные вопросы:**

1. Что такое «конвенции о связи»? Перечислите конвенции, которые вы

знаете, и уточните их содержание.

Правила, декларирующие способы передачи параметров при организации связи модулей, получили название «конвенции». Поскольку первоначально основные правила передачи управления и параметров определялись

языком программирования, названия основных конвенций связано с именами

двух основных универсальных языков программирования: Паскаль и Си. Остальные получили свои имена в соответствии с основными свойствами: стандартная Windows, защищенная и регистровая.

Конвенция Паскаль предполагает, что параметры помещаются в стек в

том порядке, в котором они встречаются в списке формальных параметров

подпрограммы. Причем все параметры передаются через стек, регистры для

передачи параметров не используются. Завершаясь, подпрограмма удаляет

параметры из стека, а потом возвращает управление.

Конвенция Си предполагает обратный порядок помещения параметров в

стек, регистры также не используются, и параметры из стека удаляет вызывающая программа.

Стандартная и Защищенная конвенции используют обратный порядок

занесения параметров в стек, но очистку стека вызываемой процедурой. Эти

конвенции очень похожи. Отличие только в том, что Защищенная конвенция

формирует исключение при обнаружении ошибок, связанных с передачей

параметров.

Регистровая конвенция означает передачу до трех параметров в регистрах. Обычно этого хватает, но если параметров больше, то остальные передаются через стек.

2. Каково внутреннее представление данных в программе на Паскале? На С++?

С++:

char – 1 байт

short – 1 байт

int – 2 байта

long – 4 байта

long long – 8 байт

Также к данным этого типа могут дописывать префикс, обозначающий наличие знака unsigned/signed

Pascal:

shortint - 1 байт

integer - 2 байта

longint - 4 байта

byte - 1 байт без знака

word - 2 байта

Переменные в обоих языках хранятся в оперативной памяти, арифметические действия происходят через регистры и часть через регистры и память

3. Какие особенности компоновки необходимо учитывать при написании модуля на ассемблере?

Как уже и говорилось существуют разные конвенции о связях, которые необходимо учитывать при программировании функций и процедур, также нужно обращать внимание на то как разные языки при ассемблировании изменяют имена переменных, что представлено в таблице 2:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Delphi Pascal | Borland C++ | Visual С++ |
| Чувствительность к регистру клавиатуры | Не различает строчных и прописных  букв | Различает строчные и прописные буквы | Различает строчные и прописные буквы |
| Преобразование внешних имен | Преобразует все строчные буквы имен в прописные | Помещает символ «\_» перед внешними именами | Помещает символ «\_» перед внешними именами |
| Преобразование имен подпрограмм | Преобразует все строчные буквы имен в прописные | Изменяет внутреннее имя подпрограммы: @<Имя>$q<описание параметров> | Изменяет внутреннее имя подпрограммы: @\_<имя>@<число параметров\* 4> |

Таблица 2 – Особенности формирования внутренних имен

4. Какие конвенции вы использовали при создании своей программы?

Конвенцию pascal

5. Как связана структура данных стека в момент передачи управления и

текст программы и подпрограмм?

На примере конвенции pascal:

Вначале в основной программе в стек передаются аргументы и адрес возврата для процедуры, затем процедура запоминает текущую позицию стека.

Процедура рассчитывает относительно базового адреса адресы операндов в стеке и работает с ними.

А после выполнения возвращает позицию стека в исходное положение удаляя переданные параметры.

6. С какой целью применяют разно языковые модули в одном проекте?

С целью использовать библиотеки кода, написанные на другом языке, если таковых нет на текущем языке. Или для оптимизации скорости работы если используется библиотека более быстро действенного языка

**Вывод:**

Я изучил конвенции о способах передачи управления и данных при вызове из программы, написанной на языке высокого уровня, подпрограмм, написанных на ассемблере. И написал программу для решения поставленной задачи, опираясь на полученные знания.