|  |  |
| --- | --- |
| Gerb-BMSTU_01 | ***«*Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

**по лабораторной работе № 5**

# Дисциплина: Машинно-зависимые языки и основы компиляции

**Название лабораторной работы:** Связь разноязыковых модулей

Студент гр. ИУ6-42Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. Д. Шатский**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ C.C.Данилюк**

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

**Вариант 2.23**

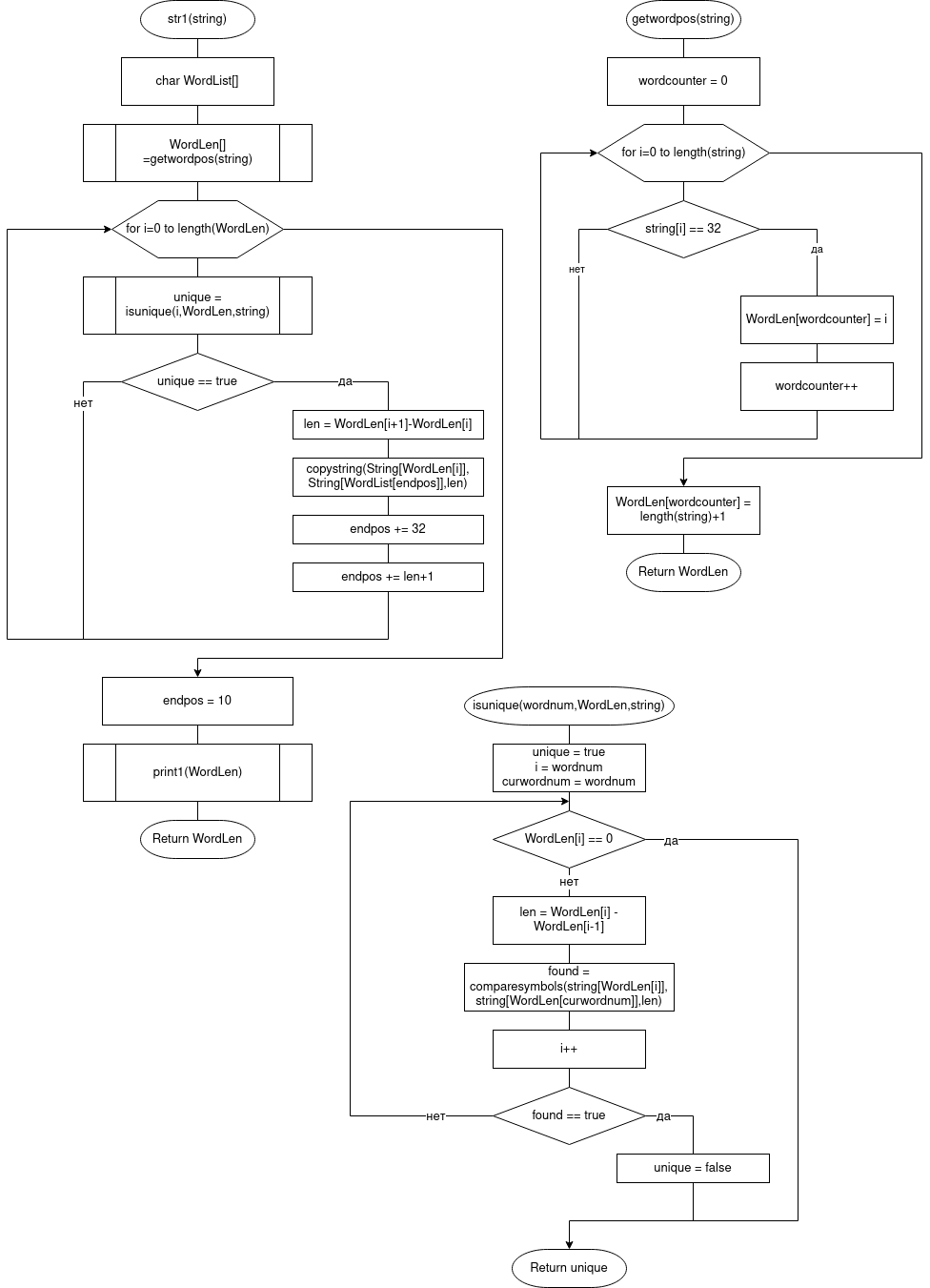
**Цель работы.** изучение конвенций о способах передачи управления и данных при вызове из программы, написанной на языке высокого уровня, подпрограмм, написанных на ассемблере.

**Задание.**

Дан текст не более 255 символов. Слова отделяются друг от друга пробелами. Составить словарь уникальных слов.

**Схема.**

Схему алгоритма можно увидеть на рисунке 1.



*Рисунок 1 – Схема алгоритма*

**Программа.**

**C++.**

#include <iostream>

#include <string.h>

extern "C++" char\* str1(const char\* n);

extern "C++" void print1(char\* n);

using namespace std;

int main() {

char string[256];

cout << "Enter string:\n";

cin.getline(string,sizeof(string));

strcat(string, " ");

str1(string);

return 0;

}

void print1(char\* output)

{

cout << "Unique words:\n";

std::cout << output << "\n";

}

**NASM.**

section .data

section .bss

WordList resb 256

WordLen resb 25

WordCount resb 1

global \_Z4str1PKc

extern \_Z6print1Pc

extern \_Z10printdebugib

section .text

\_Z4str1PKc:

push rbp

mov rbp,rsp

push rcx

push rax

push rbx

push rdx

push rdi

push rsi

push r8

push r9

push r12

push r14

mov r8,rdi

call getwordspos

mov rbx,0

mov rcx,0

mov cl,dl

cycle1:

push rcx

mov r14,0

mov rdx,0

mov cl,bl

call isunique

cmp rdx,0

jne skipadd

push rcx

push rdi

push rsi

mov cl,[WordLen+rbx+1]

sub cl,[WordLen+rbx]

dec cl

push rcx

mov r9,r8

add r9b,[WordLen+rbx]

mov RSI,r9

lea RDI,[WordList+r12]

rep movsb

pop rcx

add r12b,cl

mov BYTE[WordList+r12],32

inc r12b

pop rsi

pop rdi

pop rcx

skipadd:

pop rcx

inc rbx

dec ecx

jnz cycle1

pop r14

pop r12

pop r9

pop r8

pop rsi

pop rdi

lea rdi,[WordList]

call \_Z6print1Pc

pop rdx

pop rbx

pop rax

pop rcx

mov rsp,rbp

pop rbp

lea rax,[WordList]

ret

getwordspos:

mov r12,0

mov rbx,-1

mov rdx,1

nextsym:

inc rbx

cmp BYTE[r8+rbx],0

je after

cmp BYTE[r8+rbx],32

jne nextsym

mov [WordLen+rdx],bl

inc BYTE[WordLen+rdx]

inc rdx

inc BYTE[WordCount]

jmp nextsym

after:

mov [WordLen+rdx],bl

inc BYTE[WordLen+rdx]

ret

isunique:

push rcx

push r8

push r9

push rsi

push rdi

push rbx

push r12

mov rdx,0 ; not found

mov rbx,0

mov bl,cl

mov cl,[WordLen+rbx+1]

sub cl,[WordLen+rbx]

dec cl

push rbx

mov r12,rbx

cmpagain:

cmp BYTE[WordLen+rbx+2],0

je endcmp

mov cl,[WordLen+r12+1]

sub cl,[WordLen+r12]

dec cl

mov rsi,0

mov rdi,0

mov sil,[WordLen+rbx+1]

mov r14,r8

add rsi,r14

mov dil,[WordLen+r12]

add rdi,r14

inc cl

inc cl

rep cmpsb

cmp cl,0

jne notfound

mov rdx,1 ; found

jmp endcmp

notfound:

inc rbx

jmp cmpagain

endcmp:

pop rbx

; push rdi

; push rsi

; push rax

; push rdx

; mov rdi,rbx

; mov rsi,rdx

; call printdebug

; pop rdx

; pop rax

; pop rsi

; pop rdi

pop r12

pop rbx

pop rdi

pop rsi

pop r9

pop r8

pop rcx

ret

printdebug:

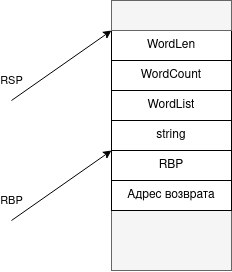
cmp [WordCount],bl

je skipprint

call \_Z10printdebugib

skipprint:

ret



*Рисунок 2 – Структура стека*

*Z:\home\explorer\Downloads\Untitled Diagram.drawio(6).png*

*Рисунок 3 – Структурная декомпозиция*

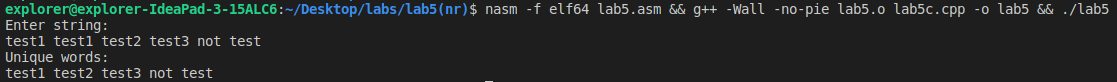
**Тесты.**

Результаты тестирования можно увидеть в таблице 1.

Таблица 1 – результаты тестирования

|  |  |  |
| --- | --- | --- |
| Исходные данные | Ожидаемый результат | Полученный результат |
| test1 test1 test2 test3 not test | test1 test2 test3 not test | test1 test2 test3 not test |
| test test test test test | test | test |
| a aaa aaaa aa a aaa a a aaa a aa | aaaa aaa a aa | aaaa aaa a aa |
| a word is a word and sentence is a sentence | word and is a sentence | word and is a sentence |
| word | word | word |

Результат запуска программы на выполнение можно увидеть на рисунке 2.



*Рисунок 4 – Результат выполнения программы*

**Контрольные вопросы.**

**1. Что такое «конвенции о связи»? В чем заключается конвенция register?**

Конвенции о связи это правила передачи параметров при организации связи модулей на различных языках.

Конвенция register заключается в том, что первые три параметра передаются через регистры EAX, EDX и ECX, а остальные передаются черезстек.

**2. Что такое «пролог» и «эпилог»? Где располагается область локальных   
данных?**

Пролог это специальные команды, которые необходимо написать в начале подпрограммы для сохранения состояния регистров до вызова функции и правильной передачи управления.

Эпилог это специальные команды, которые необходимо написать в конце подпрограммы для сохранения состояния регистров до вызова функции и правильной передачи управления.

Область локальных данных располагается в стеке.

**3. Как связана структура данных стека в момент передачи управления и   
текст программы и подпрограмм?**

В момент передачи управления структура стека зависит от конвенций о связи разноязыковых модулей. Обычно в стек помещаются передаваемые параметры, и данные необходимые для возврата управления.

**4. С какой целью применяют разноязыковые модули в одном проекте?**

С помощью вызова модуля, написанного на низкоуровневом языке программирования можно повысить скорость работы необходимой части программы, написанной на языке программирования более высокого уровня.

**Вывод.** Былиизучены конвенции о способах передачи управления и данных при вызове из программы, написанной на языке высокого уровня, подпрограмм, написанных на ассемблере.