| 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 (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 Прикладная информатика.**

**ОТЧЕТ**

| **по лабораторной работе №** | 5 |
| --- | --- |

Название: Связь разноязыковых модулей

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

| Студент |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Я.С. Петрова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Цель работы:** изучение конвенций о способах передачи управления и

данных при вызове из программы, написанной на языке высокого уровня (С++), подпрограмм, написанных на ассемблере.

**Задание**



# Ход выполнения

1) Составим схему алгоритма (рис. 1) для решения поставленной задачи:

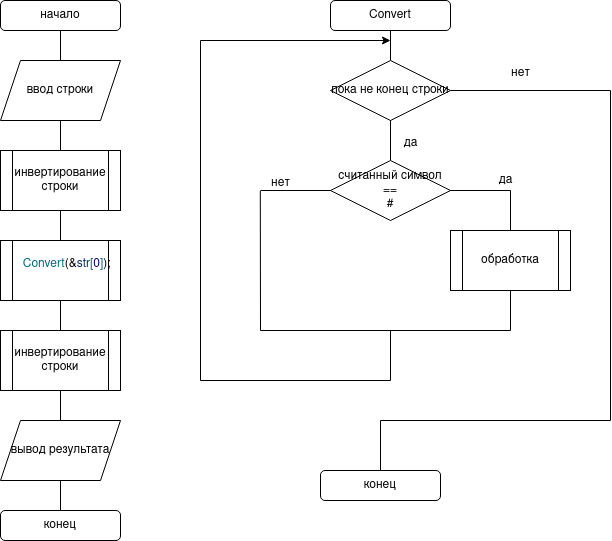


Рисунок 1 – Схема алгоритма программы

2) Напишем программу на языке С++, выполняющую ввод, вызов подпрограммы. Код программы представлен в листинге ниже.

Листинг 1 – Код программы на С++

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

extern "C" void Convert(char\*);

int main () {

string str;

cout << "Введите строку: ";

getline(*cin*, *str*);

reverse(str.begin(), str.end());

str.append(" \n");

Convert(*&str[*0*]*);

reverse(str.begin(), str.end());

str.append(" \n");

cout << str;

return 0;

}

Напишем подпрограмму на языке ассемблера с учетом регистровой конвенции.

Листинг 2 – Код подпрограммы на языке ассемблера

global Convert

section .text ; сегмент кода

Convert:

push RBP

mov RBP, RSP

mov RBX, RDI ; первая переменная - исходная строка

mov ECX, 0 ;side\_p

cycle\_usual:

mov AL, [RDI]

cmp AL, 0

je Exit

cmp AL, 23h ;сравнить символ

je prep ;Переход, если есть #

; Если нет #, идём дальше

inc RDI ; увеличиваем основной указтель

mov RBX, RDI ; обновляем сторонний указатель (для удаления)

jmp cycle\_usual

;совпадают

prep:

mov RDX, 0

mov BYTE DL, [RDI + 1] ; первый элемент последовательности

mov RCX, 1

cycle\_find:

inc RDI

mov RSI, 0

cmp BYTE [RDI], DL ;сравнить

; Выход, если не совпал

jne del

; Продолжаем, если совпал

inc RCX ; увеличиваем счётчик

jmp cycle\_find

del:

cmp RCX, 2

je cycle\_usual

jmp clear

clear:

cmp RCX, 0

je cycle\_usual

mov BYTE [RBX], 20h

inc RBX

dec RCX

jmp clear

Exit:

pop RBP

ret

Для сборки программы используется среда QtCreator. Для того, чтобы трансляция и компоновка прошла без сбоев, и не потерялась возможность отлаживать полученную программу, необходимо настроить файл проекта .pro.

Листинг 3 – Код файла проекта .pro

QT -= gui

CONFIG += c++11 console

CONFIG -= app\_bundle

# You can make your code fail to compile if it uses deprecated APIs.

# In order to do so, uncomment the following line.

#DEFINES += QT\_DISABLE\_DEPRECATED\_BEFORE=0x060000 # disables all the APIs deprecated before Qt 6.0.0

QMAKE\_EXTRA\_COMPILERS += nasm

NASMEXTRAFLAGS = -f elf64 -g -F dwarf

OTHER\_FILES += $$NASM\_SOURCES

nasm.output = ${QMAKE\_FILE\_BASE}.o

nasm.commands = nasm $$NASMEXTRAFLAGS -o ${QMAKE\_FILE\_BASE}.o ${QMAKE\_FILE\_NAME}

nasm.input = NASM\_SOURCES

NASM\_SOURCES = test.asm

SOURCES += \

main.cpp

# Default rules for deployment.

qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target

DISTFILES += \

test.asm

OBJECTS += test.o # подключение объектного модуля ассемблерной

# подпрограммы

DISTFILES += test.asm # включение в проект исходного модуля

# на ассемблере для удобства вызова его

# исходного текста в текстовый редактор

CONFIG ~= s/-O[0123s]//g # отключение оптимизации

CONFIG += -O0

Проведем трансляцию и компоновку в среде QtCreator.

3) Составим и проведём тесты программы.

Таблица 1. Сравнение ожидаемых и полученных данных.

| Исходные данные | Ожидаемый результат | Полученный результат |
| --- | --- | --- |
| adfggg# | adf |  |
| пустая строка | пустая строка |  |
| adfg | adfg |  |

Как видно из таблицы 1, все результаты тестов совпали с ожидаемыми.

# Контрольные вопросы

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

Конвенции о связи - ряд способов декларировать передачу параметров при организации связи разных модулей (вызывающей и вызываемой программы). При регистровой конвенции доступно 3 регистра при вызове подпрограммы: EAX, EDX, ECX, все остальные параметры хранятся в стеке.

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

Пролог и эпилог - стандартное оформление входа и выхода программы соответственно. При вызове программы выполняется пролог, в ходе которого в стеке сохраняется значение RBP вызывающей программы, копии параметров и область локальных переменных подпрограммы.

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

В момент получения управления подпрограммой в регистрах находятся до 6-ти параметров в виде значений или адресов (например, если передать указатель на начало строки), а в стеке – адрес возврата в вызывающую программу (адрес 64х разрядный для 64х разрядной системы).

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

Разноязыковые модули позволяют дополнить функционал языка высокого уровня. Также возможна оптимизация высоконагруженных функций, которые возможно ускорить за счет замещения стандартныой функции или метода ассемблерной программой.

**Вывод:** Я изучила конвенции о способах передачи управления и

данных при вызове из программы, написанной на языке высокого уровня (С++), подпрограмм, написанных на ассемблере.