Лабораторная работа 5 (0101 = 5)

# Модули и функции. Вызов функций стандартной библиотеки C (libc и libm)

Версия 2024 г. — засчитывается только в 2024 г. Актуальная версия — в `https://gitlab.com/illinc/gnu-asm`

**Цель работы:** изучить стандартные соглашения о вызовах и их соответствие платформам, научиться вызывать из программы на ассемблере функции стандартной библиотеки C.

Для всех заданий данной лабораторной работы программа целиком реализуется на ассемблере. Для каждого из заданий указывайте ОС, разрядность программы и соответствующее им соглашение.

Штраф за одно пропущенное обязательное задание $-2$ балла.

### Задание на лабораторную работу

**Задание Л5.з1.** Разработайте программу, выводящую на стандартный вывод группу, номер и состав команды при помощи функции $puts()$ библиотеки libc (аналогично заданию Л1.з1).

**Задание Л5.з2.** Выделите в стеке $main()$ место под переменные нескольких типов (по значению на каждый тип):
- 16-битное целое;
- 32-битное целое;
- 64-битное целое;
- 32-битное число с плавающей запятой;
- 64-битное число с плавающей запятой.

с учётом выравнивания — адрес переменной должен быть кратен как минимум её размеру. Введите в каждую из выделенных областей памяти по значению соответствующего типа при помощи $scanf()$. Напечатайте значения из памяти при помощи $printf()$. Обратите внимание, что $printf()$ и $scanf()$ имеют *переменное число аргументов,* что во многих соглашениях требует дополнительных действий.

**Задание Л5.з3.** Разработайте программу, вычисляющую по введённым значениям $x$ и $y$ с плавающей запятой двойной точности значение $z$ (таблица Л5.1), вызывая функции libm $pow()/atan2()$.

Если программа не собирается из-за отсутствия ссылок на $pow()/atan2()$, добавьте к команде сборки ключ `-lm` (указание компоновщику использовать libm).

**Задание Л5.з4. Бонус +2 балла для пар, обязательное для троек.** Реализуйте задания Л5.з2–Л5.з3 для платформы с иным соглашением о вызовах (то есть если задания Л5.з2–Л5.з3 выполнялись под 64-битной GNU/Linux — реализуйте их дополнительно и для 64-битной MS Windows, или для любой 32-битной системы).

**Варианты выражений для расчёта**

Таблица Л5.1

| $(№ - 1)\%2 + 1$ | **Вариант** |
|---|---|
| 1 | $z = \mathrm{pow}(x, y)$, $x^y$ |
| 2 | $z = \mathrm{atan2}(x, y)$, угол между вектором $(x, y)$ и осью абсцисс |

## Л5.1. Дополнительные бонусные и штрафные баллы

**—2 балла** за хранение локальных переменных в сегменте данных (использование статических/глобальных констант допускается).

**—3 балла** за нарушение соглашения о вызовах (даже если в данной конкретной программе это не имеет видимых проявлений).

## Л5.2. Ссылки на теоретические сведения

4.1. Компиляция
7.2. Типы данных
6.2. Подпрограммы и функции
6.1. Структура программы на ассемблере
5.2. Основные команды

## Л5.3. Подключение к проекту модулей на ассемблере

**Qt Creator** Файл проекта Qt Creator для добавления ассемблерного модуля необходимо отредактировать вручную, так как мастер добавления файлов не воспринимает расширения `.S` и `.s` как допустимые для исходного кода. В частности, для файла `main.S` необходимо добавить строку `SOURCES += main.S`.

**Листинг Л.1.** Файл проекта, содержащего `main.S` и `sqr.S`

```
1  TEMPLATE = app
2  CONFIG += console
3  CONFIG -= app_bundle
4  CONFIG -= qt
5
6  SOURCES += main.cpp
7  SOURCES += sqr.S
8
9  include(deployment.pri)
10 qtcAddDeployment()
```

Файлы `main.S` и `sqr.S` здесь должны находиться в той же папке, что и проект, иначе в строке после `SOURCES +=` необходимо указать имя файла с относительным путём. Других настроек, кроме редактирования файла проекта, делать не нужно.

**Code::Blocks** Создать ассемблерный модуль в Code::Blocks можно, используя меню $File \to New \to Empty\ file$. Имя файла обязательно должно иметь расширение `.S` (заглавное; расширение `.s` не воспринимается Code::Blocks как допустимое).

После создания в проекте файла с таким расширением он во время сборки проекта обрабатывается препроцессором и компилируется gcc; полученный объектный файл в дальнейшем используется компоновщиком. Дополнительных настроек делать не нужно.

### Л5.4. Вопросы

1. Какие вы знаете соглашения о вызове?
2. Какое соглашение соответствует используемой вами платформе?
3. Какая команда передаёт управление подпрограмме?
4. Какая команда возвращает управление вызывающей программе?
5. Что такое адрес возврата?