**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра математики и механики**

**ОТЧЕТ**

**по лабораторной работе № 9**

*дисциплина: Архитектура компьютера*

Студент: Кондрацкая Александра Евгеньевна

Группа: НММбд-02-24

**МОСКВА**

2024 г.

**Оглавление**

[1 Цель работы 3](#__RefHeading___Toc113_2581202437)

[2 Задание 4](#__RefHeading___Toc115_2581202437)

[3 Выполнение лабораторной работы 5](#__RefHeading___Toc117_2581202437)

[4 Выводы 19](#__RefHeading___Toc119_2581202437)

# 1 Цель работы

Цель данного задания — освоить практические навыки разработки программ с использованием подпрограмм, а также изучить основы отладки программ с помощью GDB и его ключевые функциональные возможности.

# 2 Задание

* Освоить реализацию подпрограмм в NASM
* Освоить отладку программам с помощью GDB
* Работа с данными программы GDB
* Обработать вргументы командной строки в GDB

# 3 Выполнение лабораторной работы

Создадим каталог для выполнения лабораторной работы № 9, перейдём в него и со-

здадим файл lab09-1.asm (рис. 1)

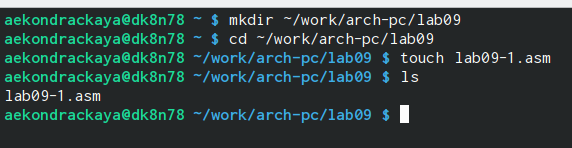


Рис. 1 Создание каталога

Введём в файл lab09-1.asm текст программы из листинга 9.1 (рис. 2)

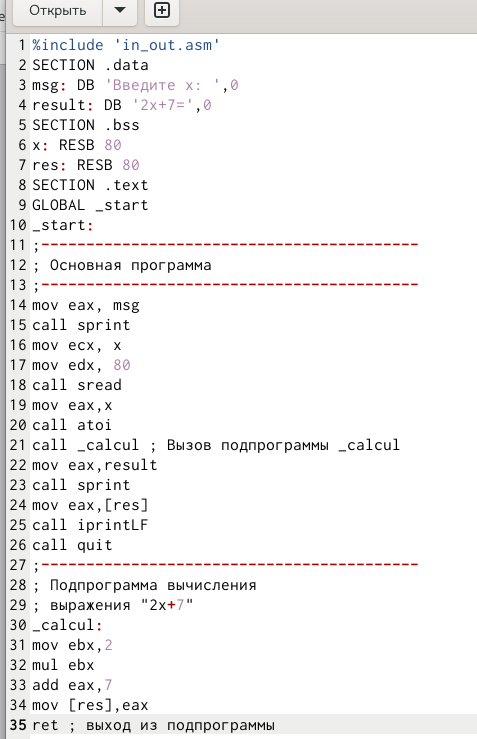


Рис. 2 Ввод программы

Создадим исполняемый файл и проверим его работу (рис. 3)

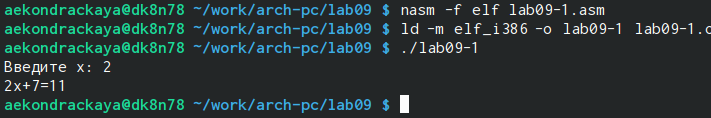


Рис. 3 Исполняемый файл

Внесем изменения в программу (рис. 4+5)

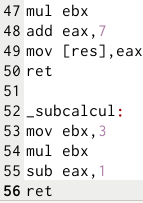
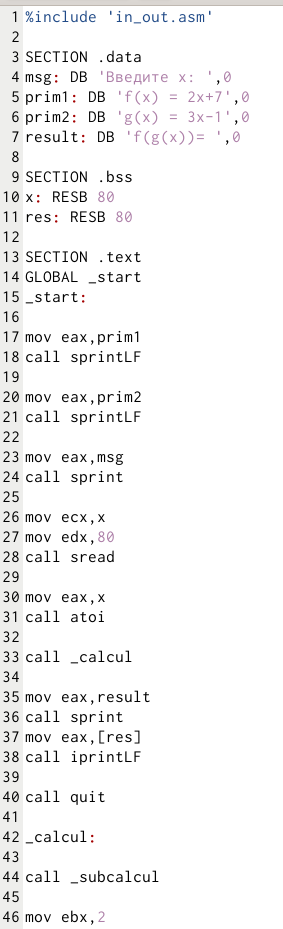


Рис. 5 Изменение в программе

Рис. 4 изменение программы

Создадим исполняемый файл и проверим (рис. 6)

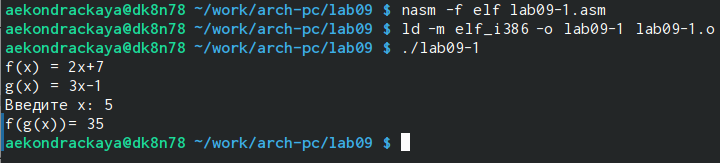


Рис. 6 Исполняемый файл

Создадим файл lab09-2.asm (рис. 7)

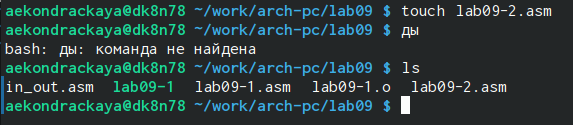


Рис. 7 Создание файла

Введем в него текст программы из листинга 9.2 (рис. 8)

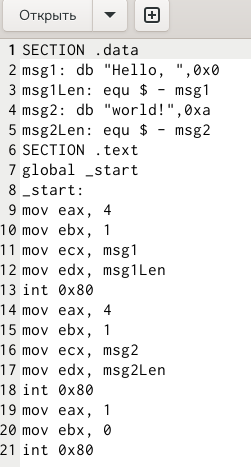


Рис. 8 Ввод программы

Создадим исполняемый файл и проверим (рис. 9)

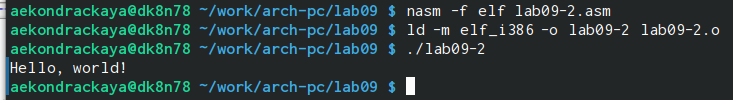


Рис. 9 Исполняемый файл

Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’. (рис. 10)



Рис. 10 Добавление отладочной информации

Загрузим исполняемый файл в отладчик gdb и проверим работу программы, запустив ее в оболочке GDB с помощью команды run (рис 11).

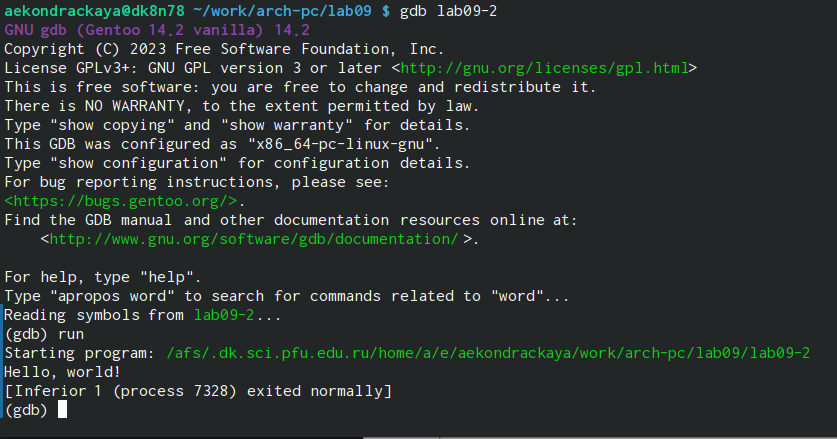


Рис. 11 Загрузка в отладчик и команда run

Для более подробного анализа программы установиv брейкпоинт на метку \_start, с

которой начинается выполнение любой ассемблерной программы, и запустиv её (рис. 12)



Рис. 12 Брейкпоинт

Посмотрим дисассимилированный код программы с помощью команды disassemble

начиная с метки \_start (рис. 13)

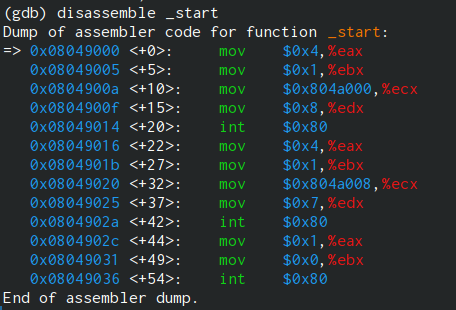


Рис. 13 Просмотр дисассимилированного кода

Переключимся на отображение команд с Intel’овским синтаксисом, введя команду set

disassembly-flavor intel (рис. 14)



Рис. 14 Переключение на другой синтаксис

Главное отличие между этими двумя способами представления команд заключается в использовании символов % и $ для обозначения регистров в дизассемблированном виде. В Intel-синтаксисе этих символов нет, что упрощает восприятие кода.

Включим режим псевдографики для более удобного анализа программы (рис. 15)

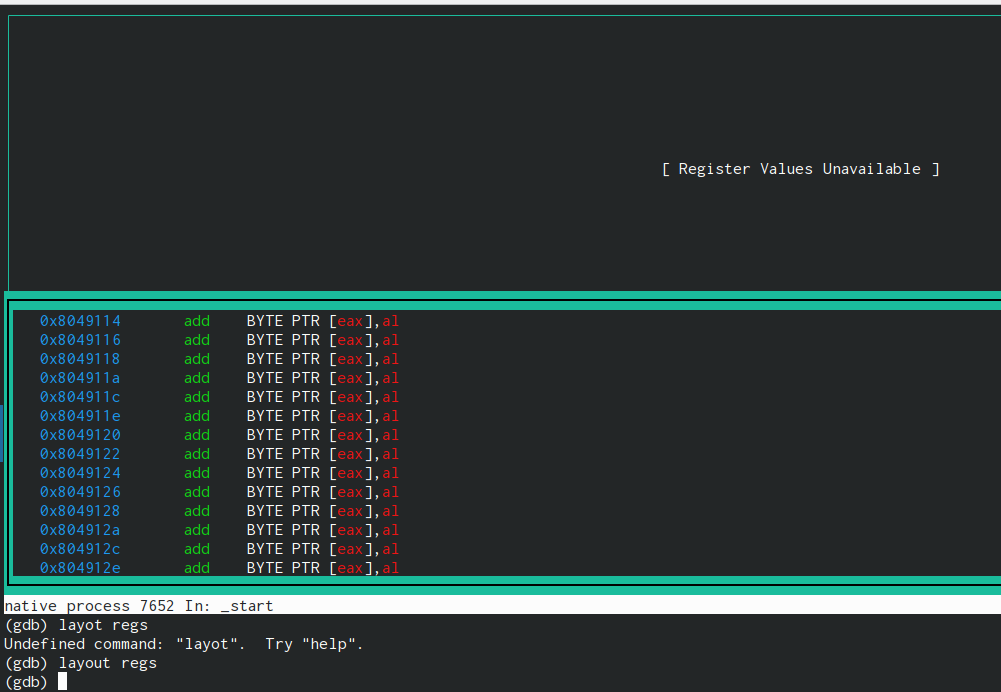


Рис. 15 Режим псевдографики

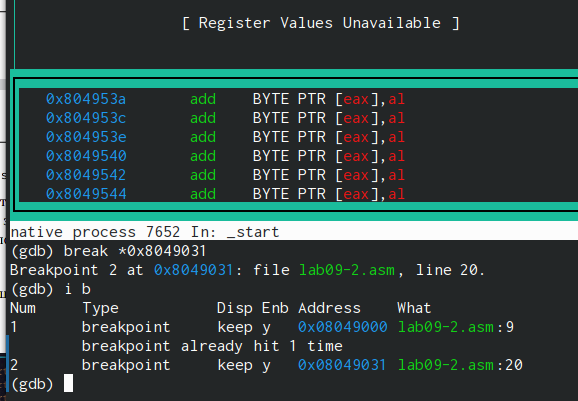
Определим адрес предпоследней инструкции (mov ebx,0x0), установим точку останова и посмотрим информацию о всех установленных точках останова (рис.16)

Рис. 16 Точки останова

Посмотрим значение переменной msg1 по имени (рис. 17)



Рис. 17 Значение переменной по имени

Посмотрим значение переменной msg2 по адресу (рис. 18)

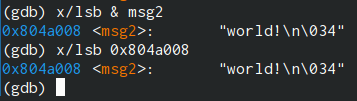


Рис. 18 Значение переменной по адресу

Изменим первый символ переменной msg1 (рис. 19)

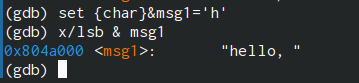


Рис. 19 Изменение символа переменной

Заменим любой символ во второй переменной msg2 (рис. 20)

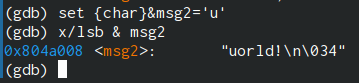


Рис. 20 Изменение символа переменной

С помощью команды set изменим значение регистра ebx (рис. 21)

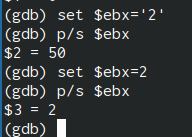


Рис. 21 Изменение значения регистра

Команда выводит два разных значения, потому что в первый раз указывается значение 2, а во второй раз регистр равен двум, поэтому и значения отличаются.

Скопируем файл lab8-2.asm, созданный при выполнении лабораторной работы №8 (рис. 22)



Рис. 22 Копирование файла

Создадим исполняемый файл (рис. 23)



Рис. 23 Исполняемый файл

Для загрузки в gdb программы с аргументами необходимо использовать ключ --args.

Загрузим исполняемый файл в отладчик, указав аргументы (рис. 24)

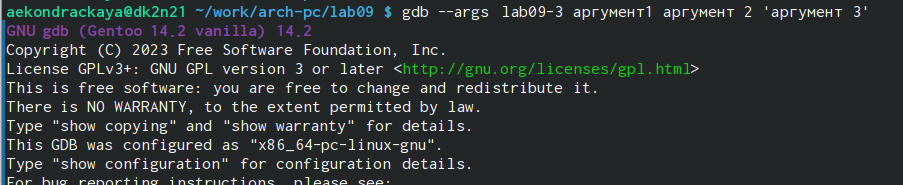


Рис. 24 Загрузка в gdb

Для начала установим точку останова перед первой инструкцией в программе и запустим ее (рис. 25)



Рис. 25 Точка останова и запуск

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число

равное количеству аргументов командной строки (включая имя программы) (рис. 26)



Рис. 26 Количество аргументов

Их ровно 5.

Посмотрите остальные позиции стека (рис. 27)

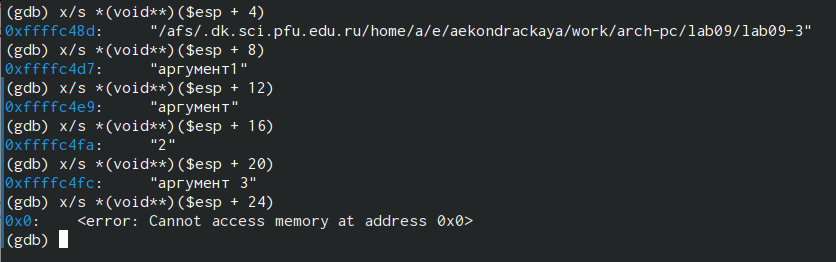


Рис. 27 Позиции стека

Первый адрес стека содержит указатель на начало данных, а следующие —   
данные с шагом 4 байта (размер элемента).Для избежания конфликтов, каждое  
значение записывается в новый стек.

САМОСТОЯТЕЛЬНАЯ РАБОТА

Создадим файл (рис. 28)

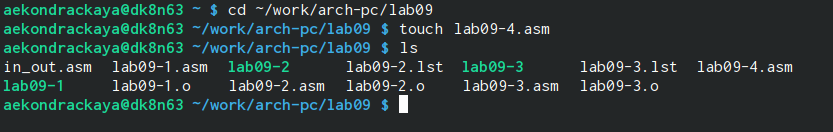


Рис. 28 Новый файл

1) Введем измененную программу (рис. 29)

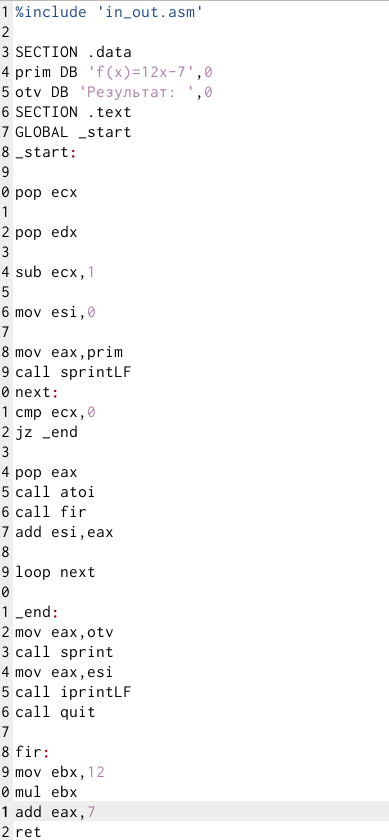


Рис. 29 Программа

Создадим исполняемый файл и проверим (рис. 30)

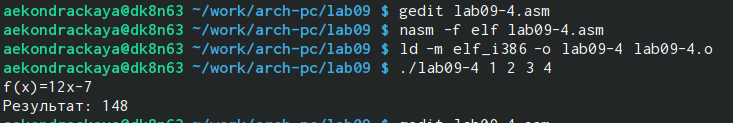


Рис. 30 Проверка

2) Создадим еще один файл (рис. 31)

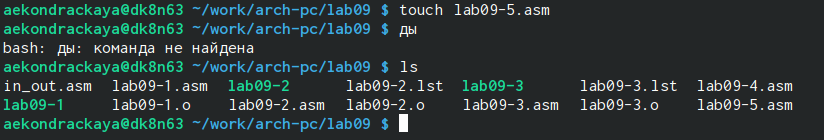


Рис. 31 Новый файл

Введем в него программу из листинга (рис. 32)

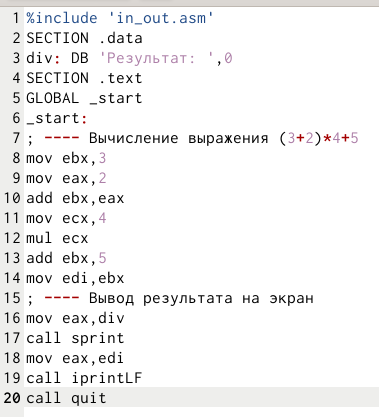


Рис. 32 Ввод программы

Создадим исполняемый файл (рис. 33)

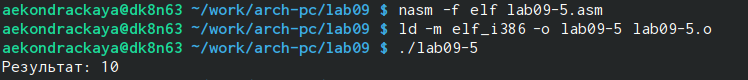


Рис. 33 Исполняемый файл

Откроем откладчик и увидим, что регистры стоят не на своих местах (рис. 34)

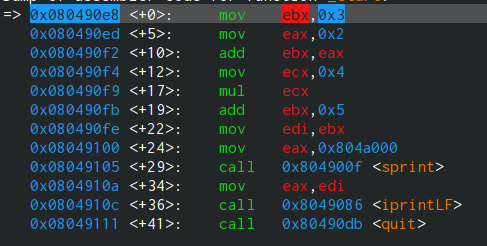


Рис. 34 Откладчик

Исправим это (рис. 35)

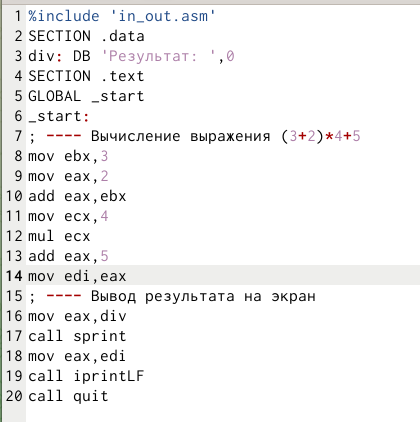


Рис. 35 Исправленная программа

Создадим исполняемый файл и проверим (рис. 36)

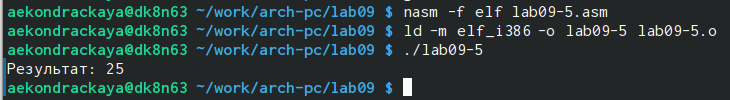


Рис. 36 Проверка

# 4 Выводы

В результате выполнения задания были приобретены навыки программирования с применением подпрограмм. Изучены базовые методы отладки программ с помощью GDB и освоены его основные функции.