Отчёт по лабораторной работе №9

Дисциплина: архитектура компьютера

Аннаоразов Сердар Аннаоразович

Содержание

# 1 Цель работы

Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

# 2 Задание

1. Реализация подпрограмм в NASM
2. Отладка программам с помощью GDB
3. Добавление точек останова
4. Работа с данными программы в GDB
5. Обработка аргументов командной строки в GDB
6. Задание для самостоятельной работы

# 3 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа: • обнаружение ошибки; • поиск её местонахождения; • определение причины ошибки; • исправление ошибки. Можно выделить следующие типы ошибок: • синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; • семантические ошибки — являются логическими и приводят к тому, что программа запускается, отрабатывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль). Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга. Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы. Последний этап — исправление ошибки. После этого при повторном запуске пр

# 4 Выполнение лабораторной работы

## 4.1 Реализация подпрограмм в NASM

Создал каталог для выполнения лабораторной работы № 9, перешел в него и создал файл lab09-1.asm:

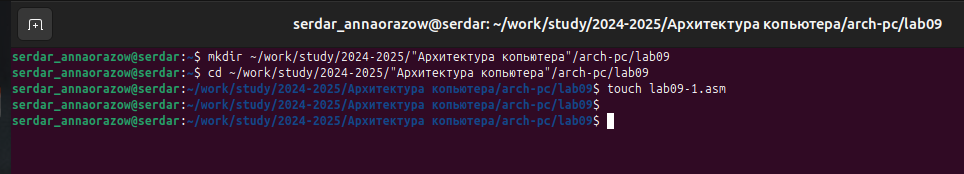


Рис. 1: Создания каталога и файла

В качестве примера рассмотрел программу вычисления арифметического выражения f(x) = 2x + 7 c помощью подпрограммы \_calcul. В данном примере x вводится с клавиатуры, а само выражение вычисляется в подпрограмме. Написал программу в Файл

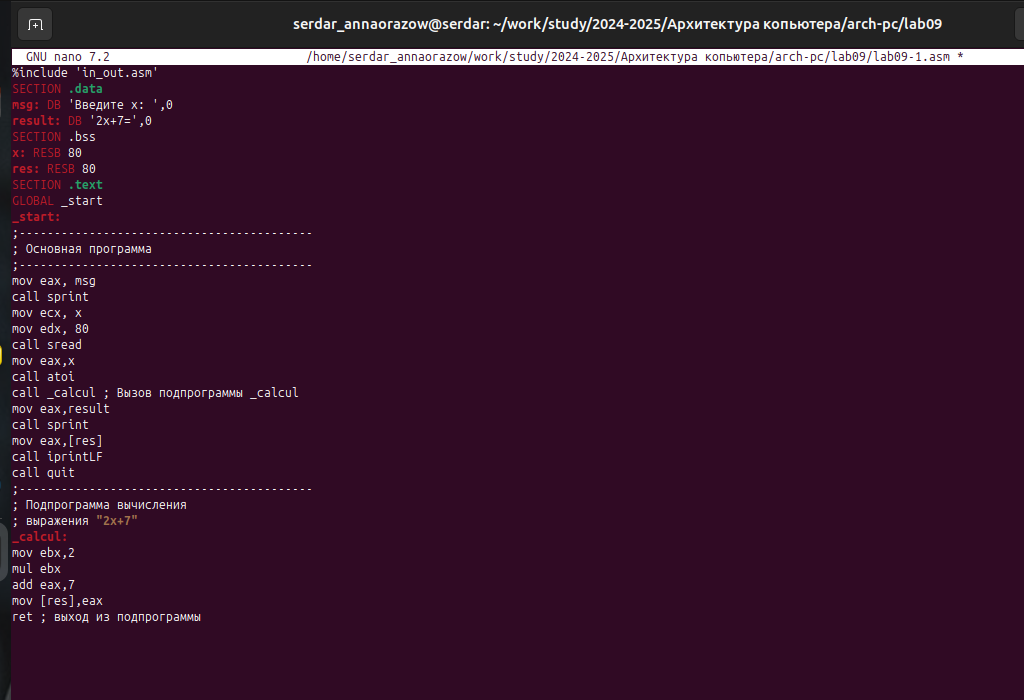


Рис. 2: Введения программы в файл

Создал исполняемый файл и запустил его

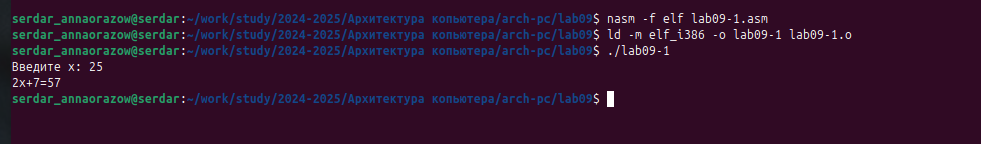


Рис. 3: Создание исполняемого файла и запуск его

## 4.2 Отладка программам с помощью GDB

Я создал новый файл lab09-2.asm

Рис. 4: Создание файла lab09-2.asm

Рис. 4: Создание файла lab09-2.asm

ввел туда программу печати сообщения Hello world!



Рис. 5: Введения программы печати сообщения Hello world

Я создал исполняемый файл и запустил его

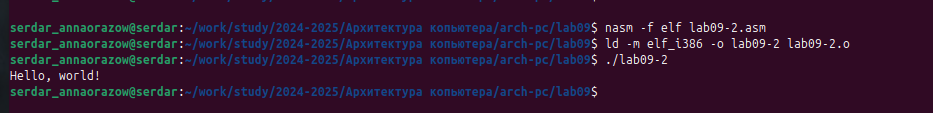


Рис. 6: Создание исполняемого файла lab09-2.asm и запуск его

Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’.Загрузил исполняемый файл в отладчик gdb

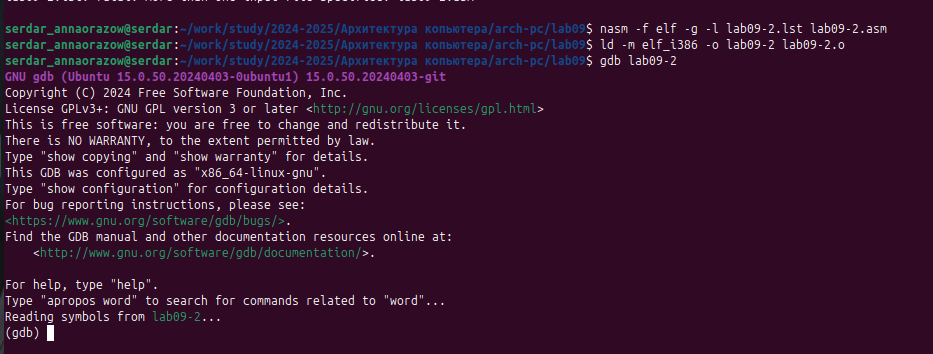


Рис. 7: Загрузка исполняемого файла в отладчик gdb

Проверил работу программы, запустив ее в оболочке GDB с помощью команды run

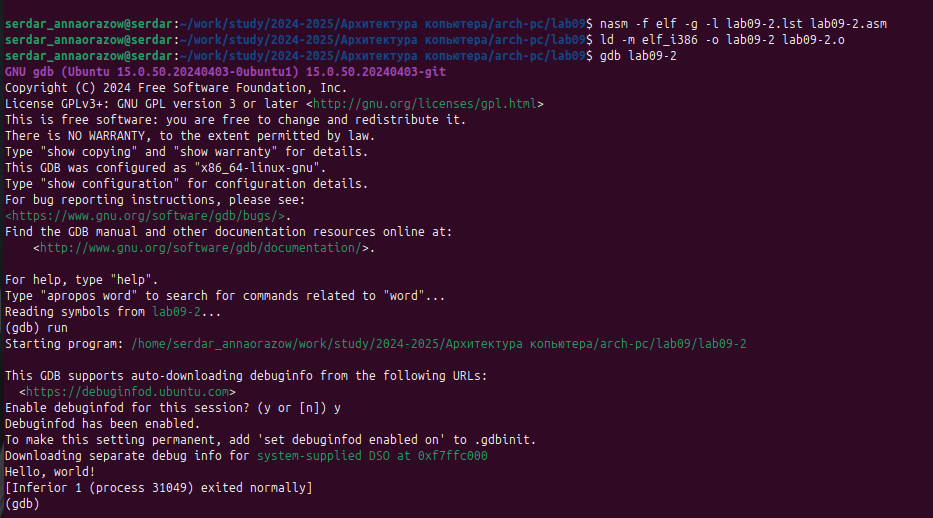


Рис. 8: проверил работу программы

Для более подробного анализа программы установил брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запустил её.

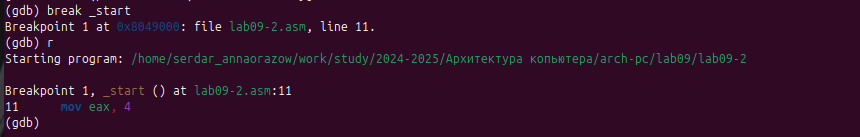


Рис. 9: установка брейкпоинт на метку \_start

Посмотрел дисассимилированный код программы с помощью команды disassemble начиная с метки \_start

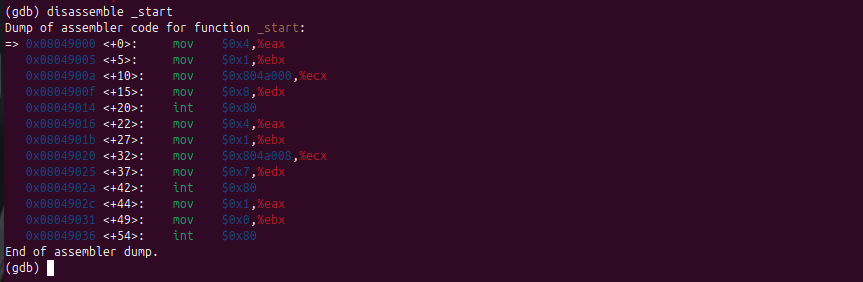


Рис. 10: Просмотр дисассимилированный код программы

Переключился на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel

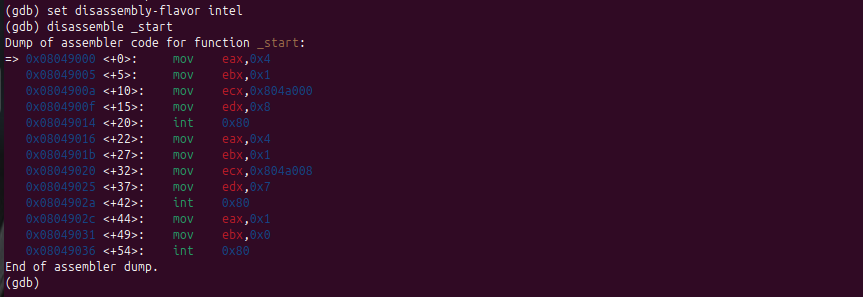


Рис. 11: Переключения на отображение команд с Intel’овским синтаксисом

Перечислил различия отображения синтаксиса машинных команд в режимах ATT и Intel.Включил режим псевдографики для более удобного анализа программы

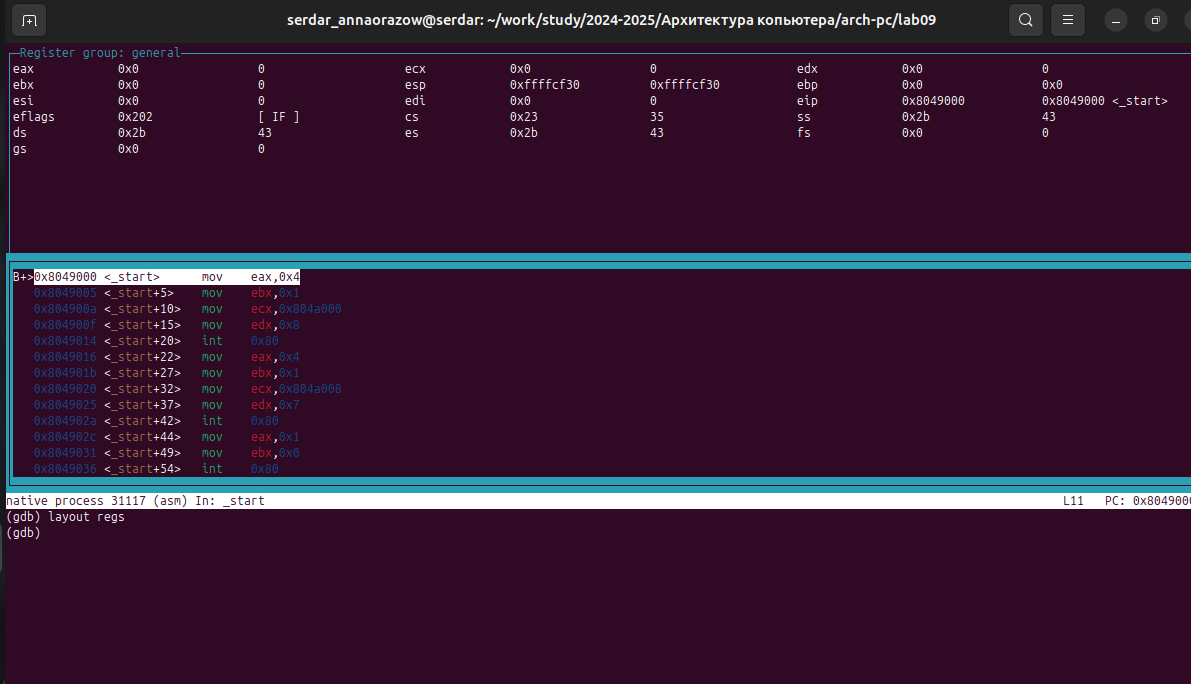


Рис. 12: режим псевдографики

Проверил установленные точки сотанова с помощью комманды info breakpoints. Установил еще одну точку останова по адресу инструкции. Адрес инструкции смог увидеть в средней части экрана в левом столбце соответствующей инструкции . Определил адрес предпоследней инструкции (mov ebx,0x0) и установил точку останова.потом посмотрел информацию о всех установленных точках останова:

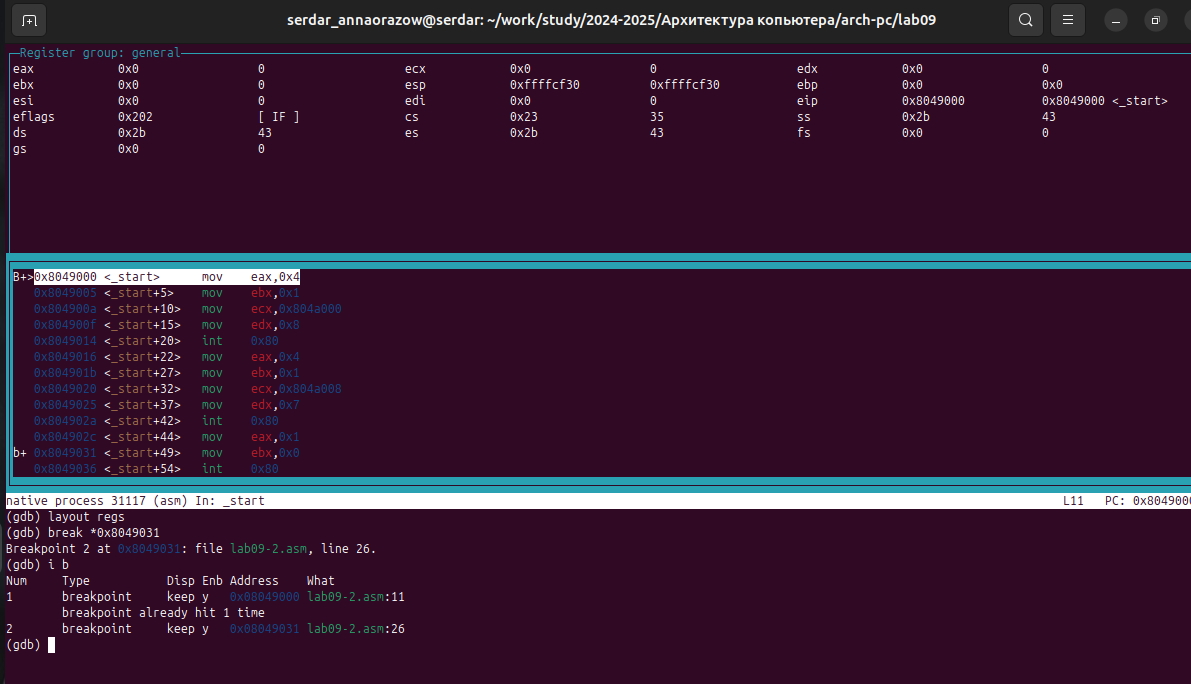


Рис. 13: Установка и Проверка точек сотанова

## 4.3 Работа с данными программы в GDB

Для начала посмотрел содержимое регистров с помощью команды info registers

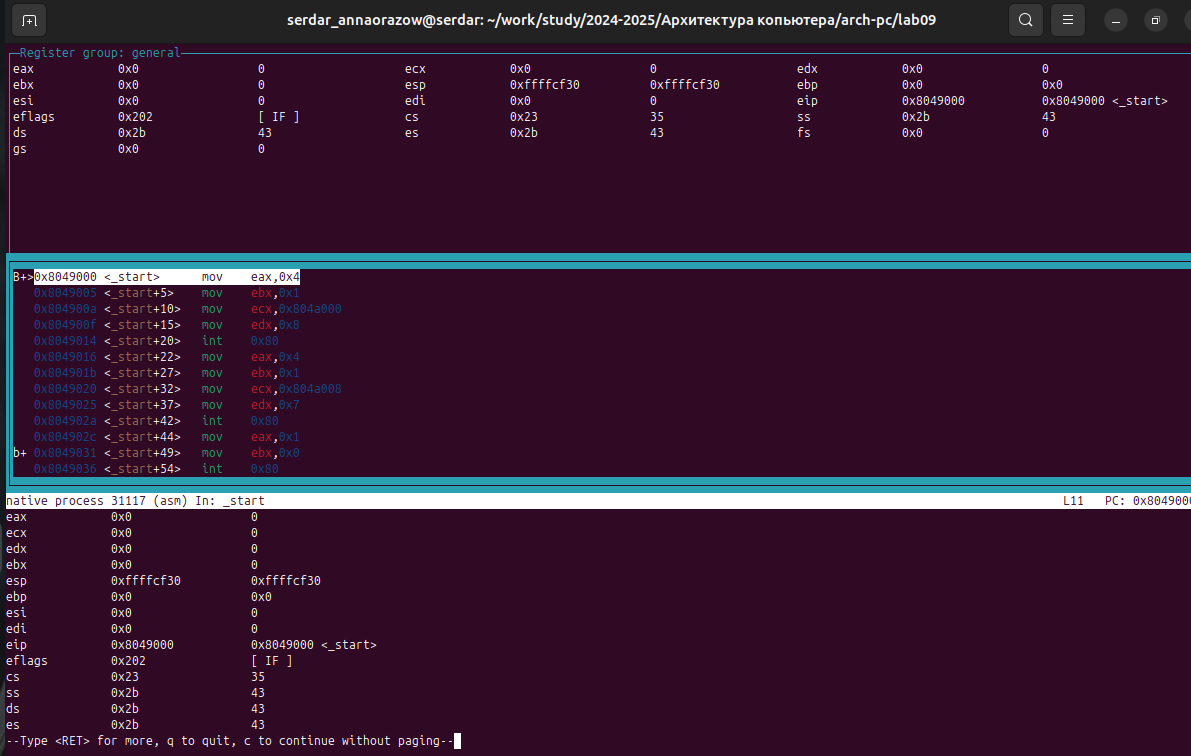


Рис. 14: Просмотр содержимое регистров

С помощью команды si я посмотрел регистры и изменил их.

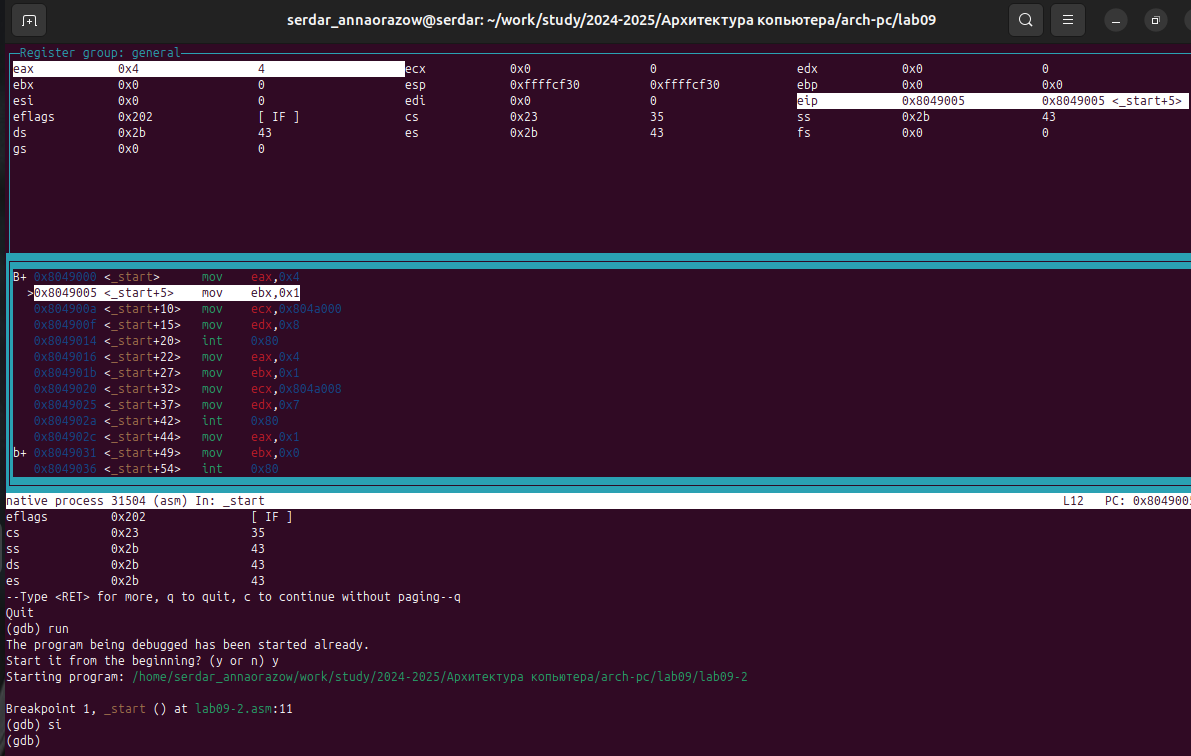


Рис. 15: Просмотр значение и изменения региястра

С помощью команды я посмотрел значение переменной msg1.

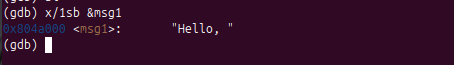


Рис. 16: Просмотр значение переменной msg1

Следом я посмотрел значение второй переменной msg2

Рис. 17: Просмотр значение переменной msg2

Рис. 17: Просмотр значение переменной msg2

С помощью команды set я изменил значение переменной msg1.

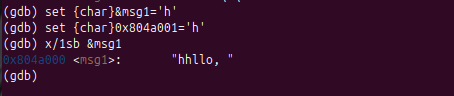


Рис. 18: Изменения значения переменной msg1

Я изменил переменную msg2

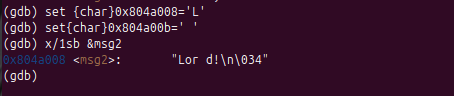


Рис. 19: Изменения переменной msg2

Я вывел значение регистров ecx и eax.

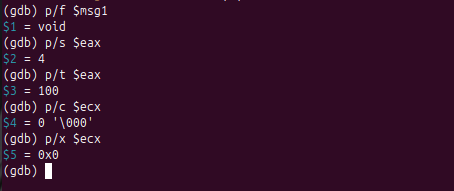


Рис. 20: Вывод значения регистров ecx и eax

Я изменил значение регистра ebx. Команда выводит два разных значения так как в первый раз мы вносим значение 2, а во второй раз регистр равен двум, поэтому и значения разные.



Рис. 21: Изменения значения регистра ebx

Я завершил работу с файлов вышел

Рис. 22: Выход

Рис. 22: Выход

## 4.4 Обработка аргументов командной строки в GDB

Я скопировал файл lab9-2.asm и переименовал его. Я сделал это через MC так как по мне это удобнее всего

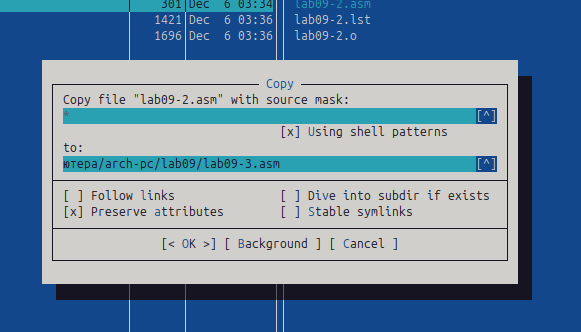


Рис. 23: Копирования файла

Запустил файл в отладчике и указал аргументы.

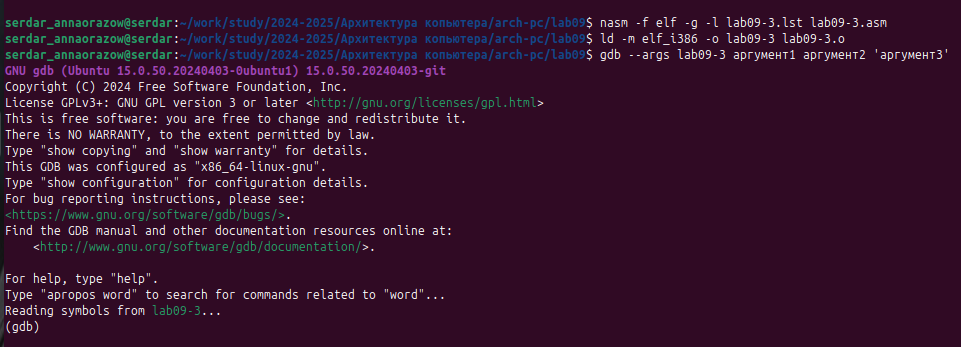


Рис. 24: Запуск файла с аргументами

Поставил метку на \_start и запустил файл

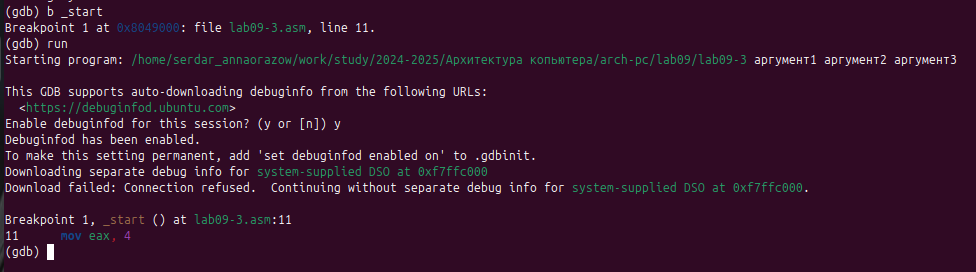


Рис. 25: Запуск файла через метку

Я проверил адрес вершины стека и убедился что там хранится 5 элементов.

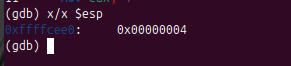


Рис. 26: Адрес вершины стека

Я посмотрел все позиции стека. По первому адрему хранится адрес, в остальных адресах хранятся элементы. Элементы расположены с интервалом в 4 единицы, так как стек может хранить до 4 байт, и для того чтобы данные сохранялись нормально и без помех, компьютер использует новый стек для новой информации.

# 5 Самостоятельная работа

## 5.1 Первая задача

Я с начала копировал файл в котором делал самомтоятельную работу 8-ой лабораторной работы. Я это сделал с помощью MC.

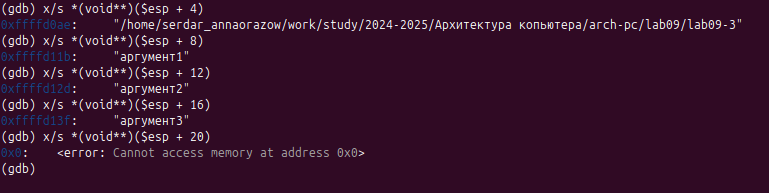


Рис. 27: Копирования файла ил lab08

Потом я преобразовал программу из лабораторной работы №8 и реализовал вычисления как подпрограмму.

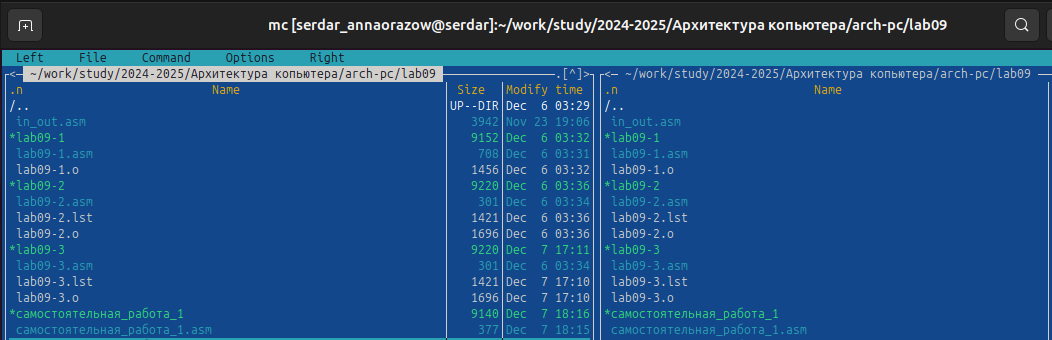


Рис. 28: Преоброзования файла

## 5.2 Вторая задача

Я создал новый файл и ввел туда программу который был в задаче. После этого я сохдал исполняемый файл и запустил его чтоб увидеть в чем ошибка.

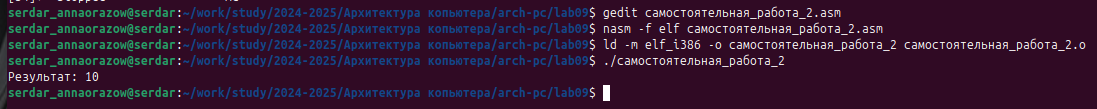


Рис. 29: Запуск файла

После проявление ошибки я запустил программу в окладчике.

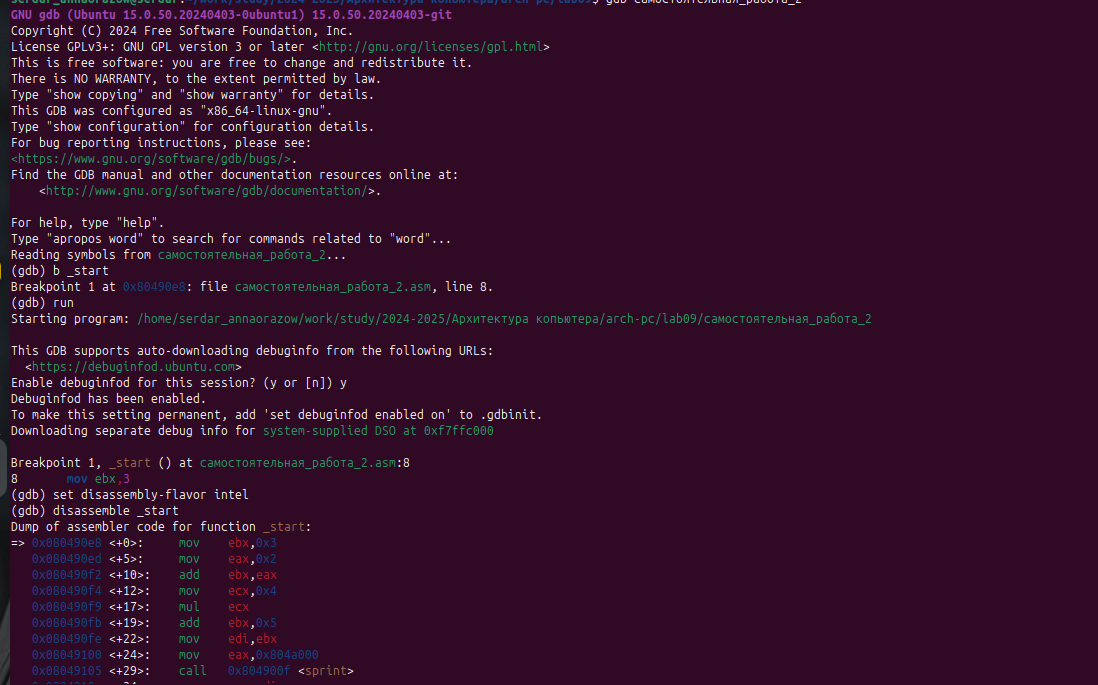


Рис. 30: Запуск программы в окладчике

Я открыл регистры и проанализировал их, понял что некоторые регистры стоят не на своих местах и исправил это.

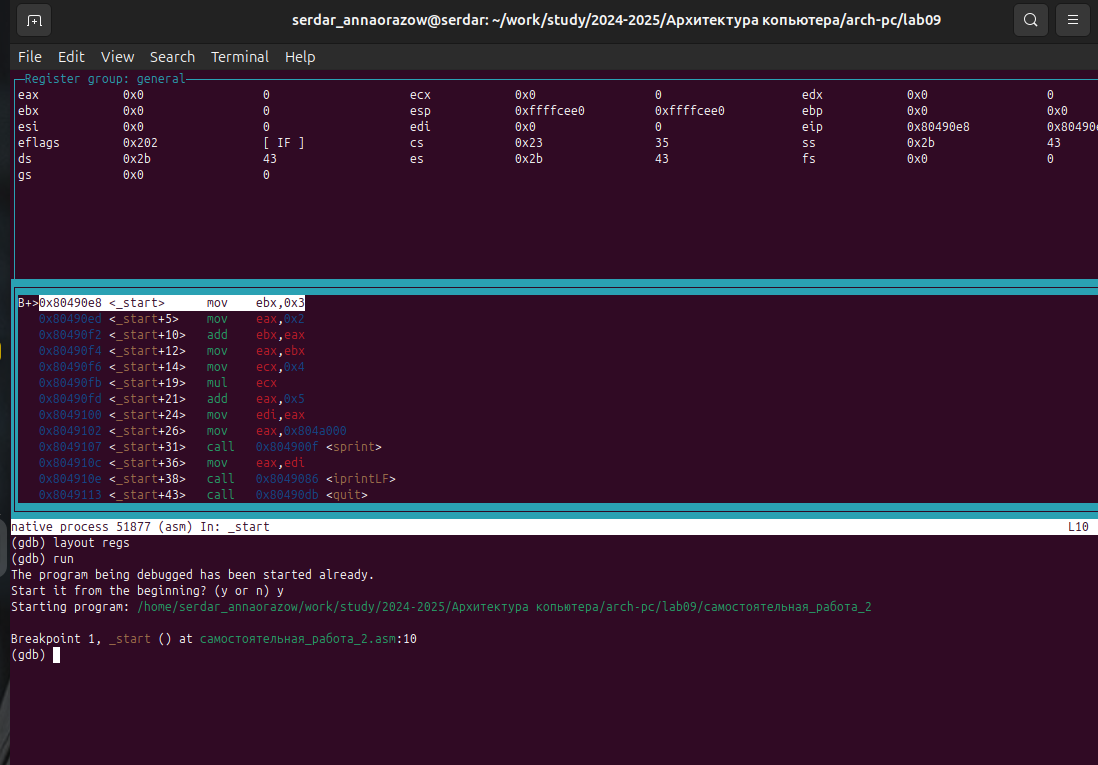


Рис. 31: Исправления ошибок

После этого я занова создал исполняемый файл и запустил его чтобы проверить что все сделано правильвно.

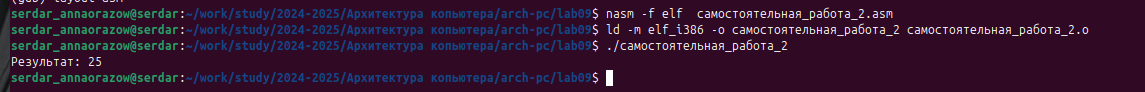


Рис. 32: Запуск исполняемого файла

# 6 Выводы

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием подпрограмм, а так же познакомился с методами отладки при поомщи GDB и его основными возможностями.

# Список литературы

1. https://esystem.rudn.ru/pluginfile.php/2089096/mod\_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F %D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0 %E2%84%969. %D0%9F%D0%BE%D0%BD%D1%8F%D1%82%D0%B8%D0%B5 %D0%BF%D0%BE%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B. %D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D1%87%D0%B8%D0%BA .pdf