Отчёт по лабораторной работе №9

Дисциплина: Архитектура компьютера

Кирьянова Екатерина Андреевна

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. 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 программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново. Наиболее часто применяют следующие методы отладки: создание точек контроля значений на входе и выходе участка программы (например, вывод промежуточных значений на экран — так называемые диагностические сообщения); использование специальных программ-отладчиков. Отладчики позволяют управлять ходом выполнения программы, контролировать и изменять данные. Это помогает быстрее найти место ошибки в программе и ускорить её исправление. Наиболее популярные способы работы с отладчиком — это использование точек останова и выполнение программы по шагам. Пошаговое выполнение — это выполнение программы с остановкой после каждой строчки, чтобы программист мог проверить значения переменных и выполнить другие действия. Точки останова — это специально отмеченные места в программе, в которых программаотладчик приостанавливает выполнение программы и ждёт команд. GDB (GNU Debugger — отладчик проекта GNU) работает на многих UNIX-подобных системах и умеет производить отладку многих языков программирования. GDB предлагает обширные средства для слежения и контроля за выполнением компьютерных программ. Отладчик не содержит собственного графического пользовательского интерфейса и использует стандартный текстовый интерфейс консоли. Однако для GDB существует несколько сторонних графических надстроек, а кроме того, некоторые интегрированные среды разработки используют его в качестве базовой подсистемы отладки.

# 4 Выполнение лабораторной работы

## 4.1 Реализация подпрограмм в NASM

Создаю каталог lab09 и файл lab9-1.asm (рис. 1).

Рис. 1: Создание каталога и файла

Рис. 1: Создание каталога и файла

Ввожу текст листинга в файл (рис. 2) и запускаю программу (рис. 3).

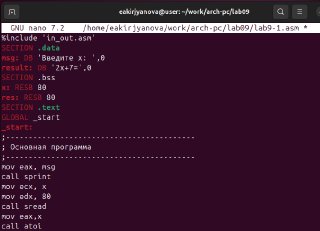


Рис. 2: Текст программы

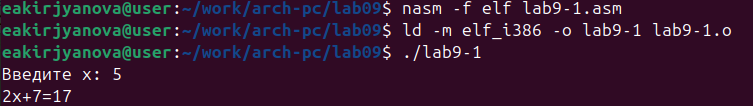


Рис. 3: Работа программы

Меняю текст программы (рис. 4), чтобы она решала выражение f(g(x)) (рис. 5).

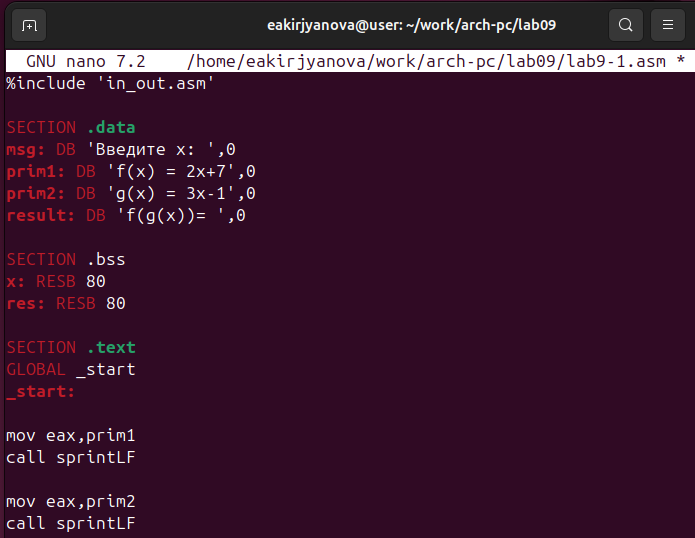


Рис. 4: Измененный текст программы

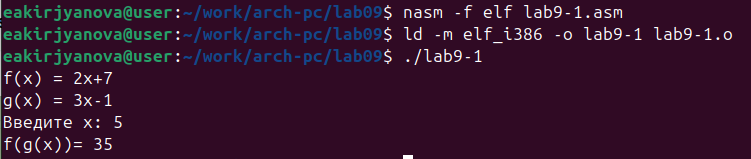


Рис. 5: Проверка работы программы

## 4.2 Отладка программам с помощью GDB

Создаю файл lab9-2.asm и ввожу туда программу (рис. 6).

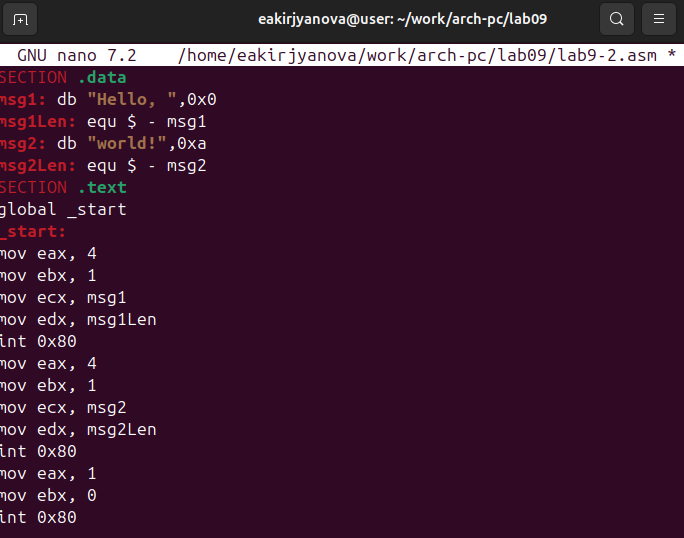


Рис. 6: Текст второй программы

Запускаю файл второй программы в отладчик gdb (рис. 7).

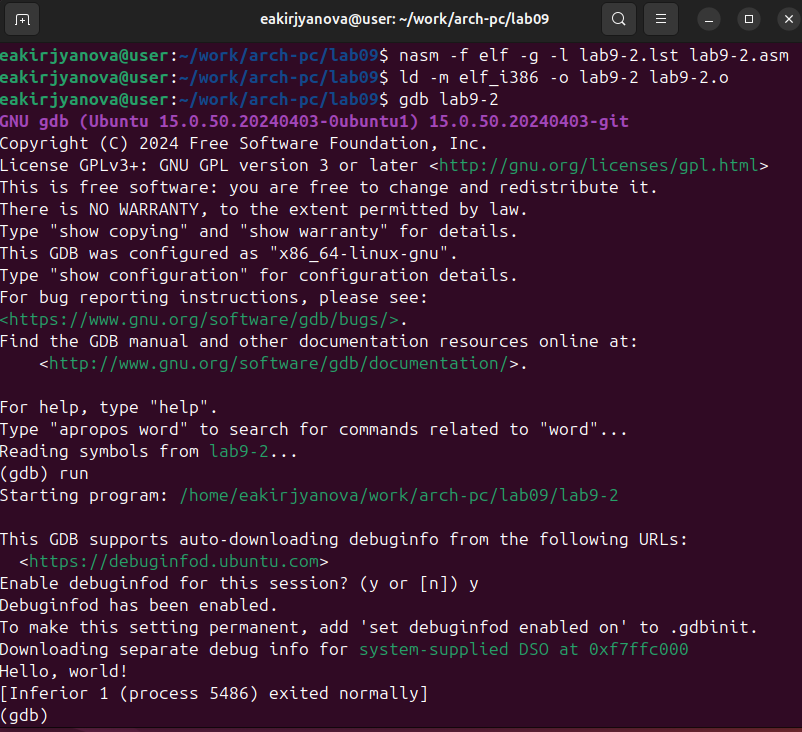


Рис. 7: Отладка второго файла

Ставлю брекпоинт на метку \_start и запускаю программу (рис. 8).

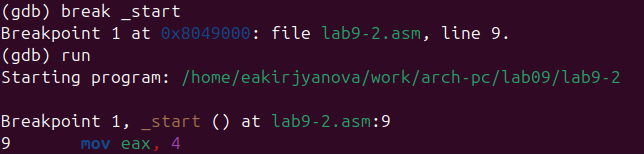


Рис. 8: Брекпоинт на метку \_start

Просматриваю дисассимплированный код программы начиная с метки (рис. 9).



Рис. 9: Дисассимплированный код

С помощью команды переключаюсь на intel’овское отображение синтаксиса (рис. 10). Отличие заключается в командах, в диссамилированном отображении в командах используют % и $, а в Intel отображение эти символы не используются. На такое отображение удобнее смотреть.



Рис. 10: Intel’овское отображение

Включаю режим псевдографики (рис. 11).

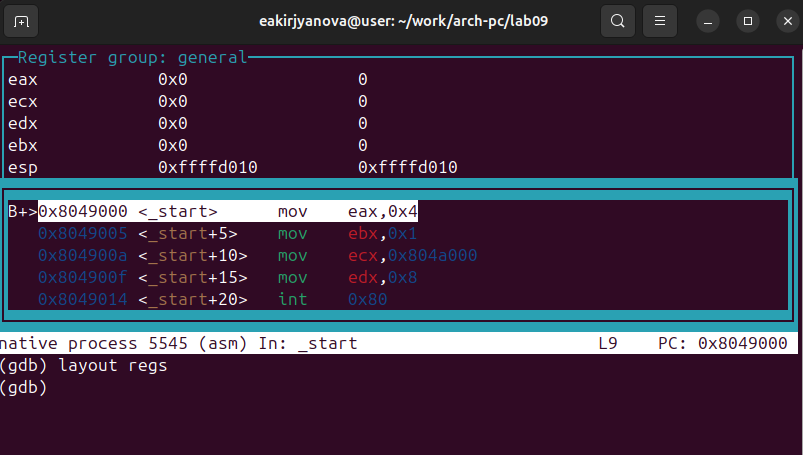


Рис. 11: Псевдографика

Смотрю наличие меток и добавляю еще одну метку на предпоследнюю инструкцию (рис. 12).

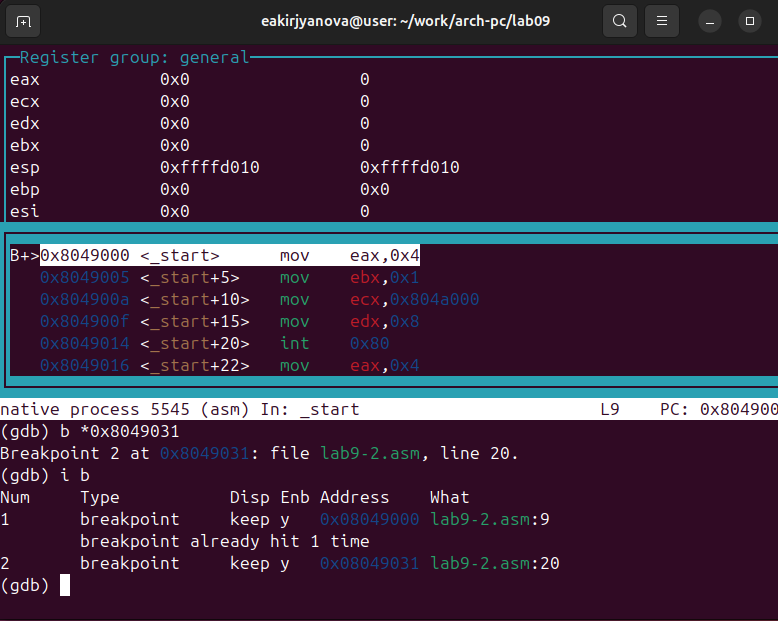


Рис. 12: Наличие меток

С помощью команды si смотрю регистры (рис. 13).

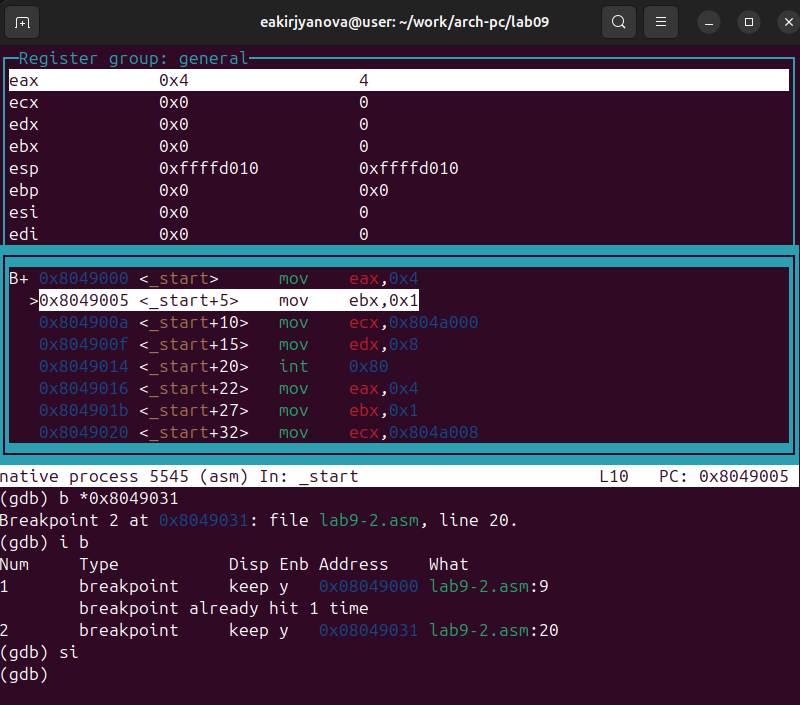


Рис. 13: Просмотр регистров

С помощью команды я смотрю значение переменной msg1 (рис. 14).

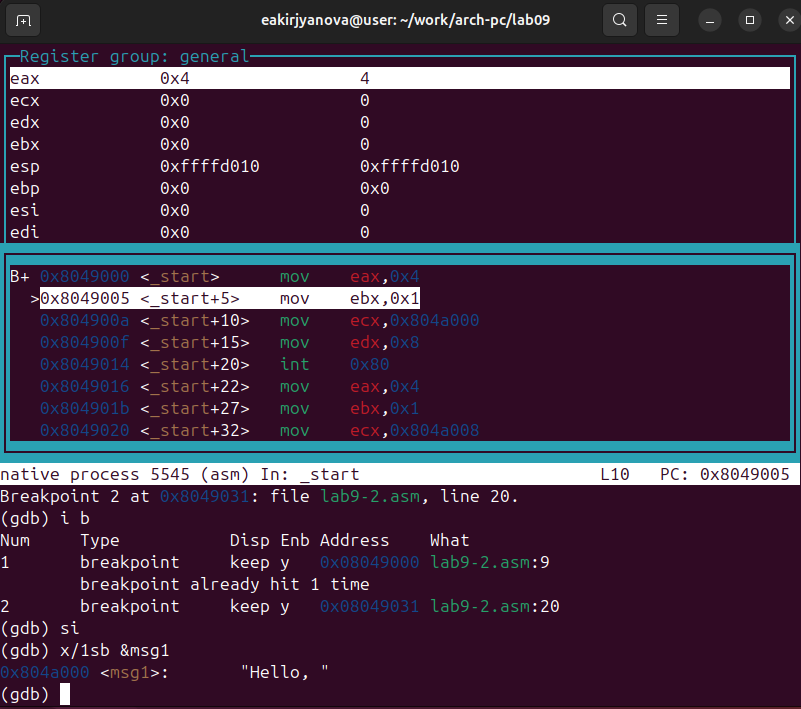


Рис. 14: Просмотри значения переменной

Смотрю значение второй переменной msg2 (рис. 15).

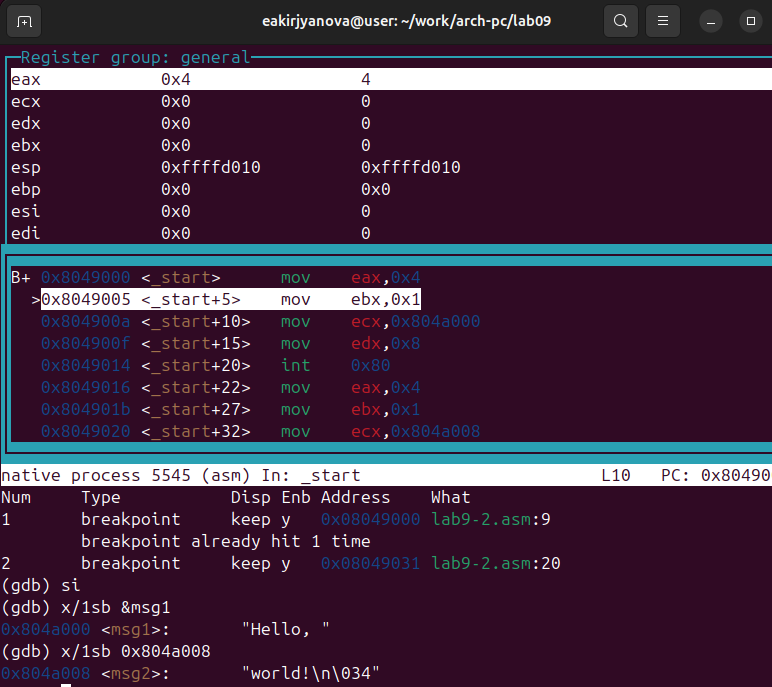


Рис. 15: Значение переменной msg2

С помощью команды set меняю значение переменной msg1 (рис. 16).

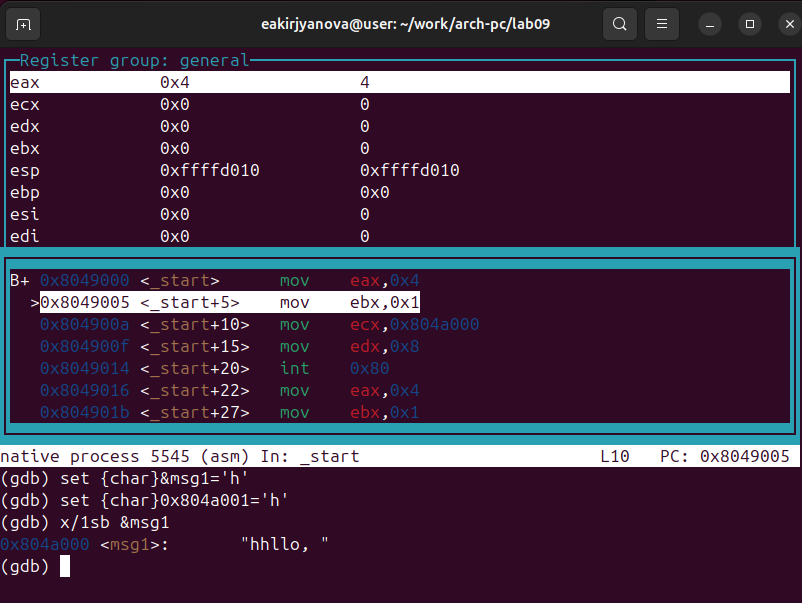


Рис. 16: Изменение значения переменной

Меняю переменную msg2 (рис. 17).

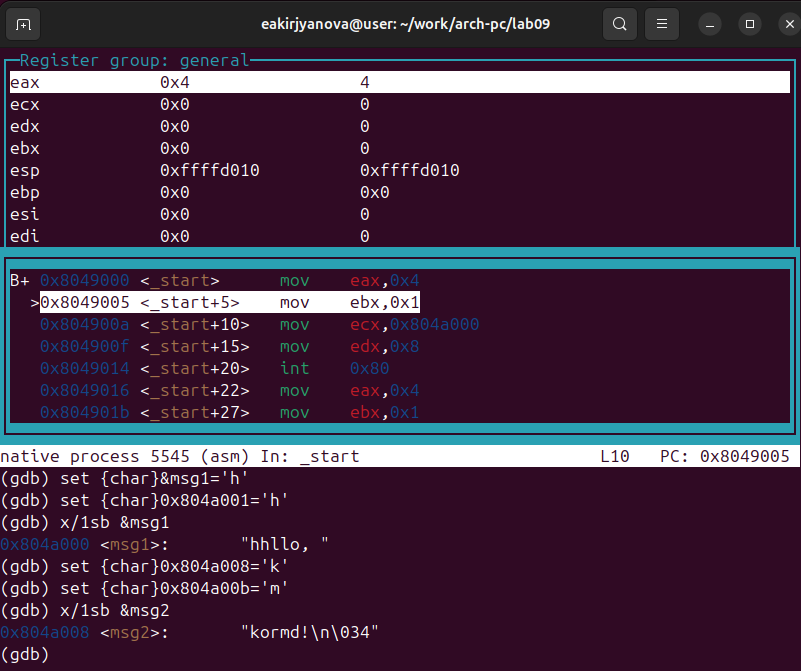


Рис. 17: Изменение msg2

Вывожу значение регистров ecx и eax (рис. 18).

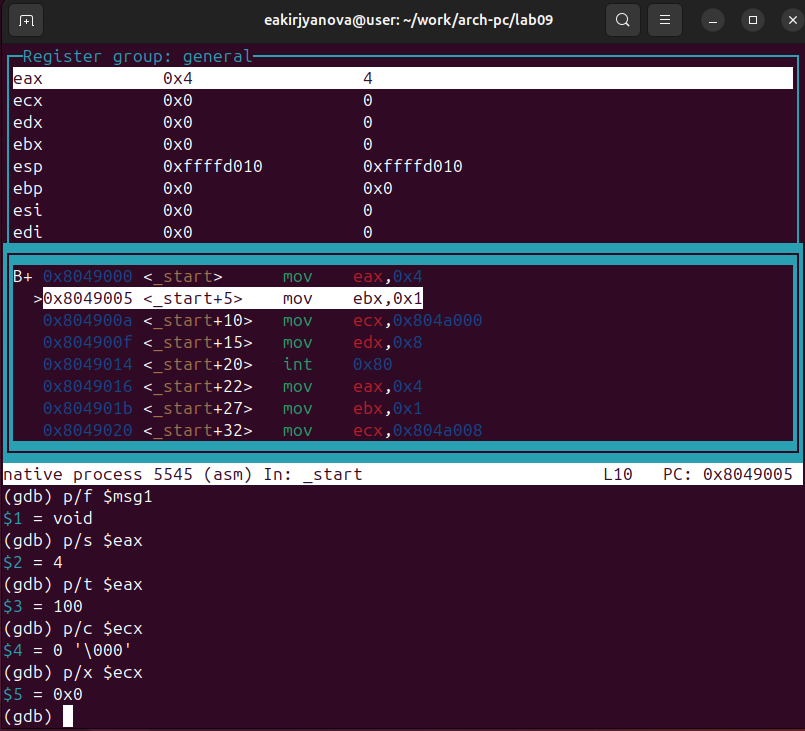


Рис. 18: Значение регистров ecx и eax

Меняю 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. 19).

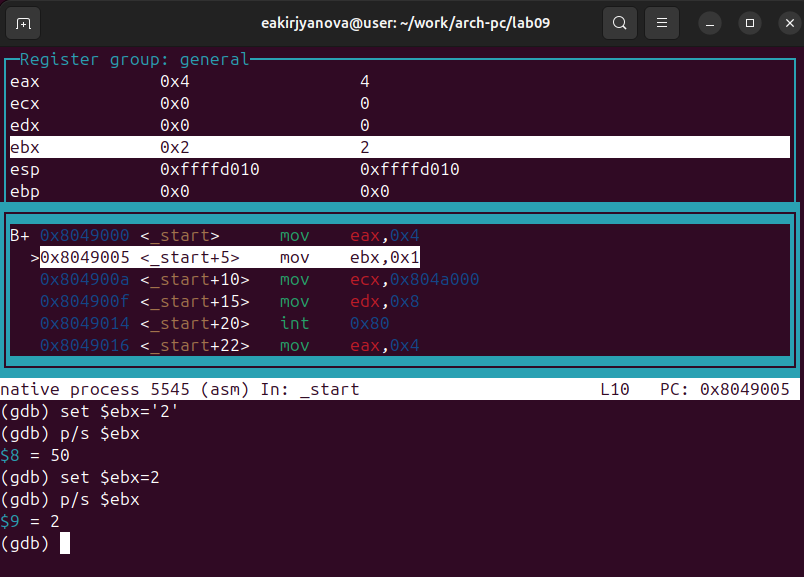


Рис. 19: Значение регистров ebx

Копирую файл lab8-2.asm и переименовываю его (рис. 20). Запуcкаю файл в отладчике, указав аргументы (рис. 21).

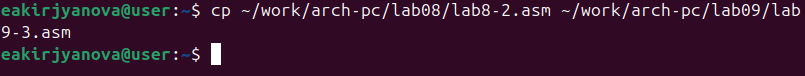


Рис. 20: Копирование

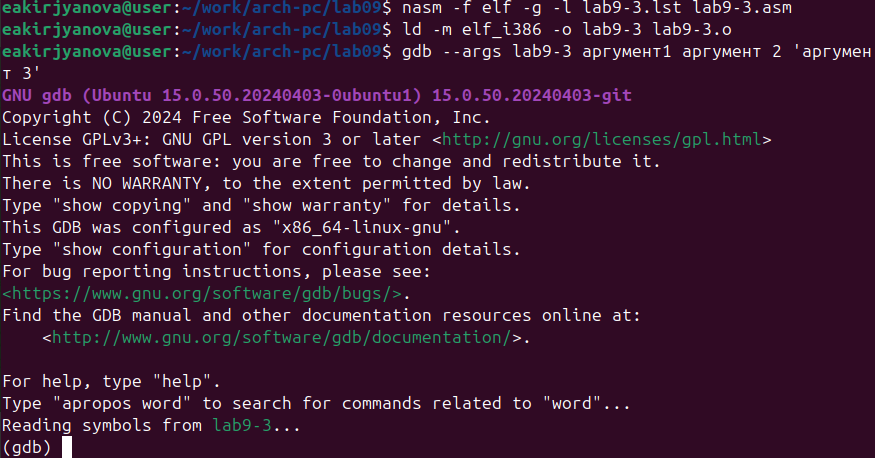


Рис. 21: Запуск файла в отладчике

Ставлю метку на \_start и запускаю файл (рис. 22).

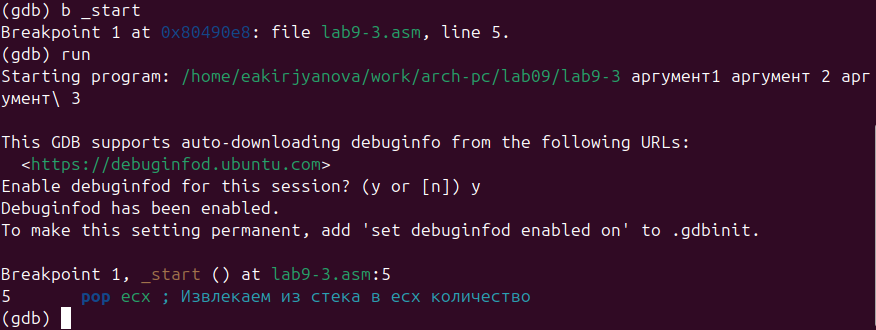


Рис. 22: Запуск файла lab9-3 через метку

Проверяю адрес вершины стека и убеждаюсь, что там хранится 5 элементов (рис. 23).

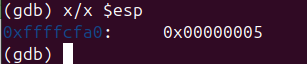


Рис. 23: Адрес вершины стека

Смотрю 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 (рис. 24).

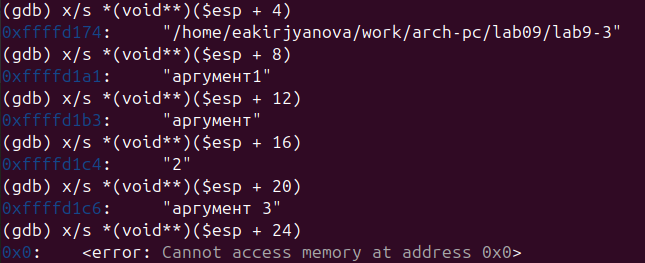


Рис. 24: Все позиции стека

## 4.3 Задание для самостоятельной работы

Преобразовываю программу из лабораторной работы №8 (рис. 25) и реализую вычисления как подпрограмму (рис. 26).

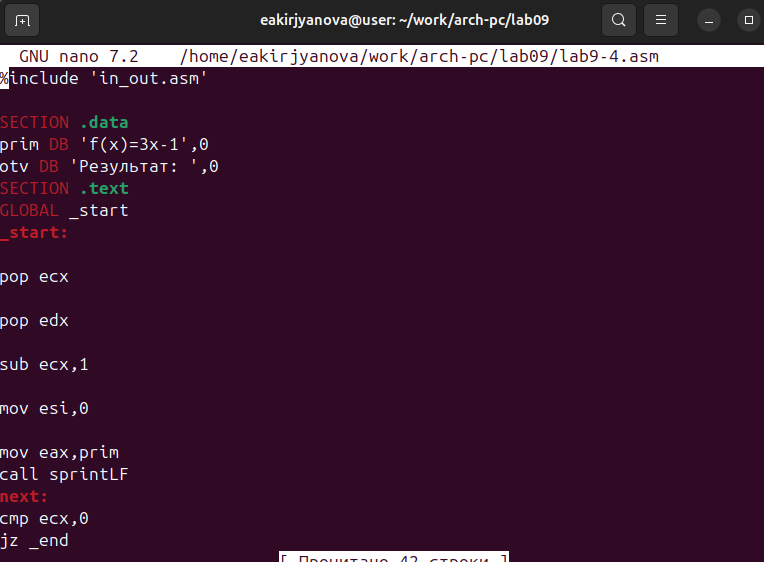


Рис. 25: Текст программы

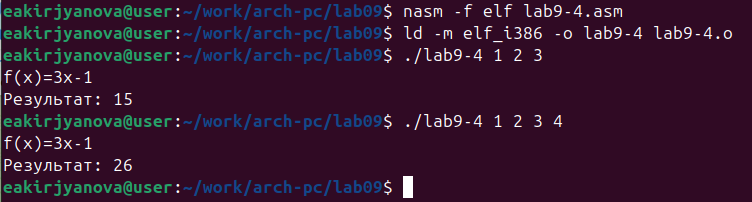


Рис. 26: Запуск программы

Переписываю программу (рис. 27) и пробую запустить ее чтобы увидеть ошибку. Ошибка арифметическая (рис. 28).

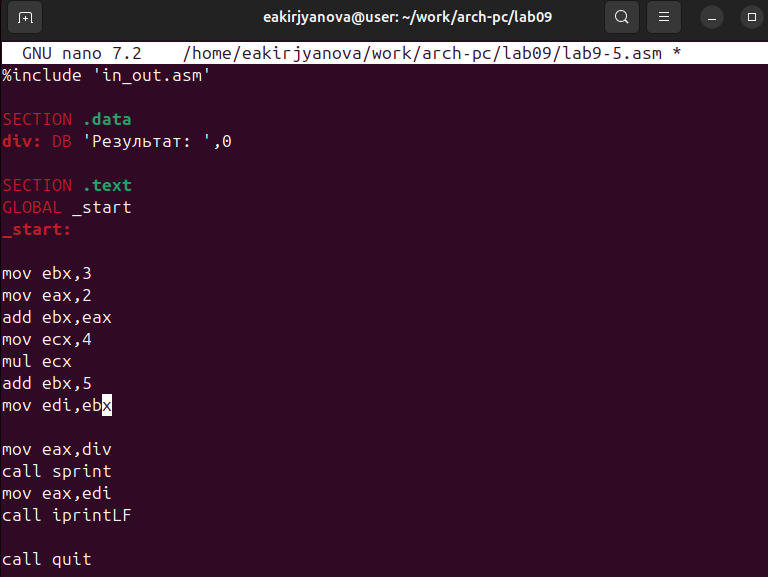


Рис. 27: Текст програмыы

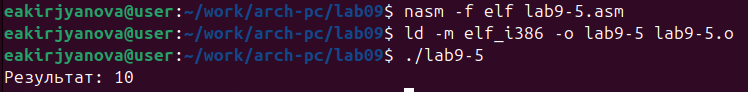


Рис. 28: Запуск программы

После появления ошибки, запускаю программу в отладчике (рис. 29).

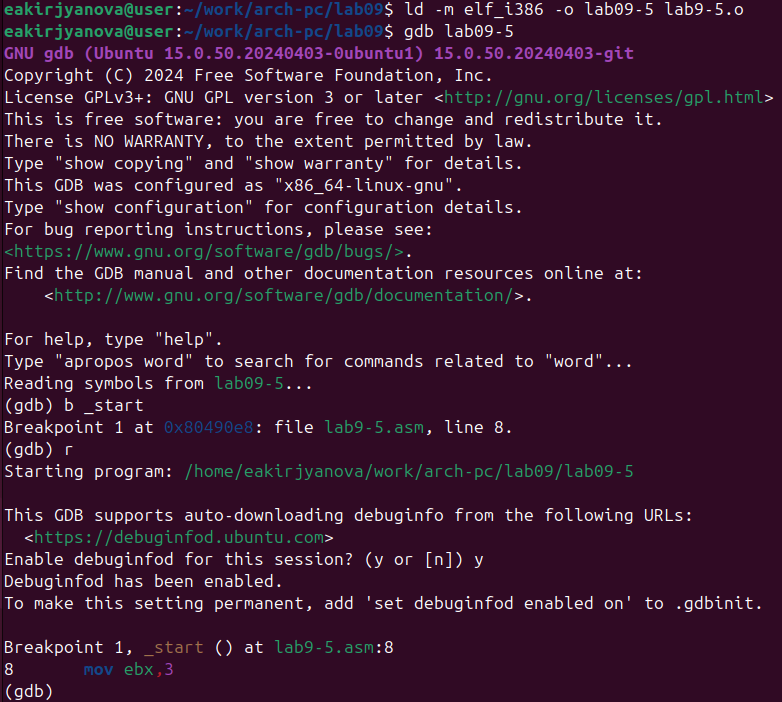


Рис. 29: Запуск программы в отладчике



Рис. 30: Запуск

Открываю регистры и анализирую их. Некоторые регистры стоят не на своих местах, исправляю это (рис. 31).

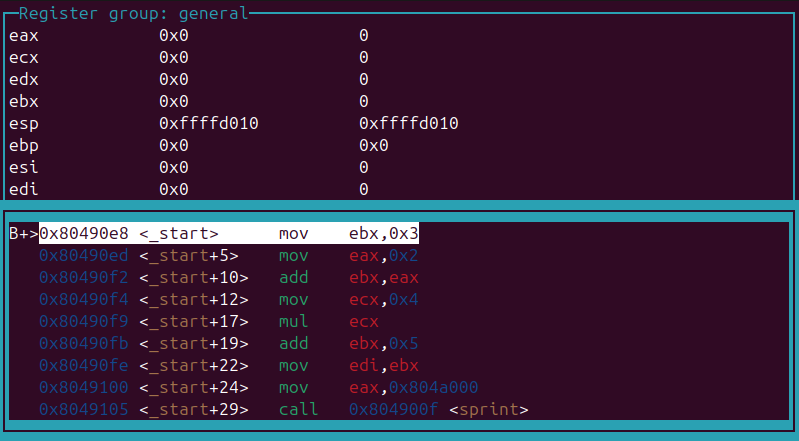


Рис. 31: Анализ регистров

Меняю регистры и запускаю программу, программа работает корректно (рис. 32).

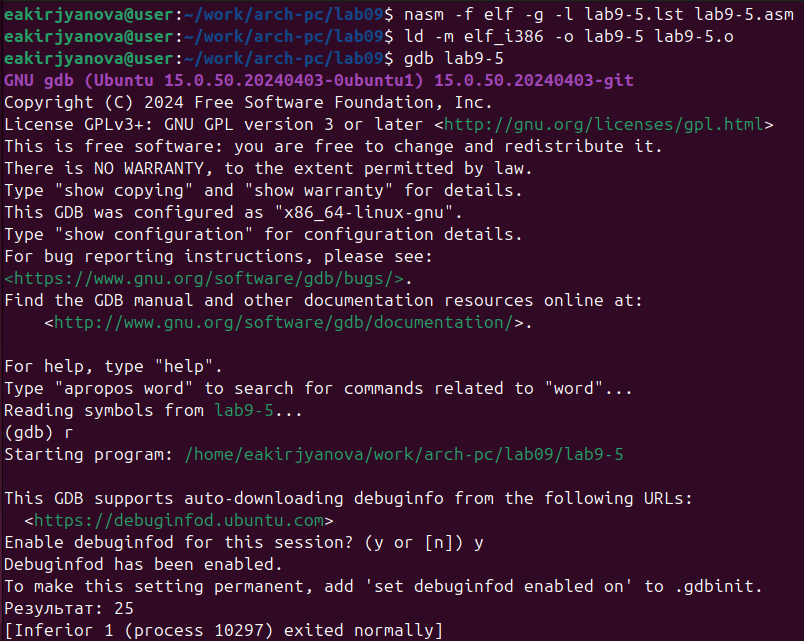


Рис. 32: Повторный запуск программы

# 5 Вывод

В ходе выполнения данной лабораторной работы я приобрела навыки написания программ использованием подпрограмм. Познакомилась с методами отладки при помозь GDB и его основными возможностями.