Отчет по лабораторной работе №9

Дисциплина: архитектура компьютера

Луцкая Алиса Витальевна

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 программ с помощью GDB
3. Задания 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 пре- рывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль).

Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить доволь- но 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.

# 4 Выполнение лабораторной работы

## 4.1 Релазиация подпрограмм в NASM

Создаю каталог для выполнения лабораторной работы № 9, перехожу в него и со- здаю файл lab09-1.asm (рис. -fig. 1).

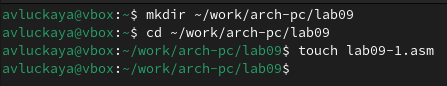


Рис. 1: Создание файла

Ввод в файл кода из листинга (рис. -fig. 2).

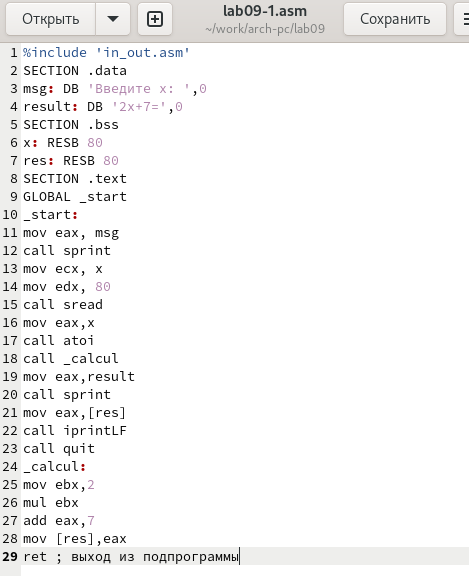


Рис. 2: Ввод программы

Компилирую и запускаю его, данная программа выполняет вычисление функции(рис. -fig. 3).

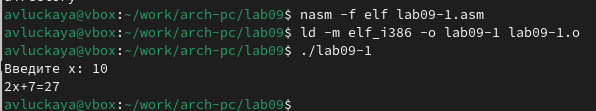


Рис. 3: Запуск программы из листинга

Изменяю текст программы, добавив в нее подпрограмму, теперь она вычисляет значение функции для выражения f(g(x)) (рис. -fig. 4).

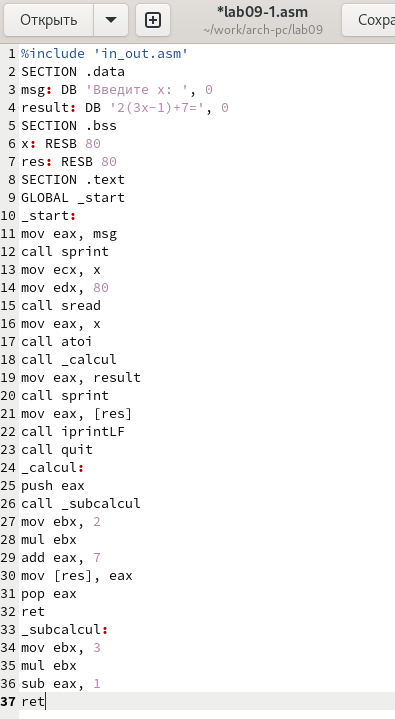


Рис. 4: Изменение программы первого листинга

Компилирую и запускаю его (рис. -fig. 5).

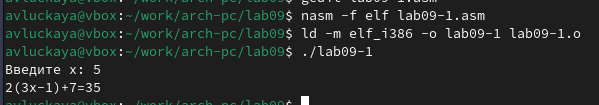


Рис. 5: Запуск программы из листинга

### 4.1.1 Отладка программ с помощью GDB

Создайте файл lab09-2.asm с текстом программы из Листинга 9.2.(рис. -fig. 6).

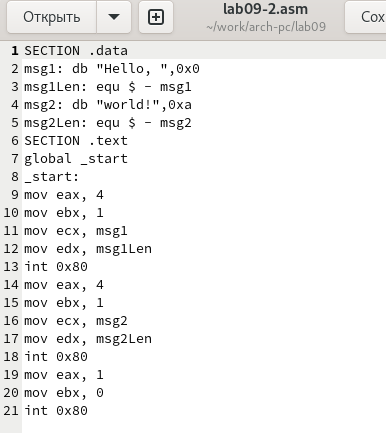


Рис. 6: Создание файла

Загружаю исполняемый файл в отладчик gdb (рис. -fig. 7).

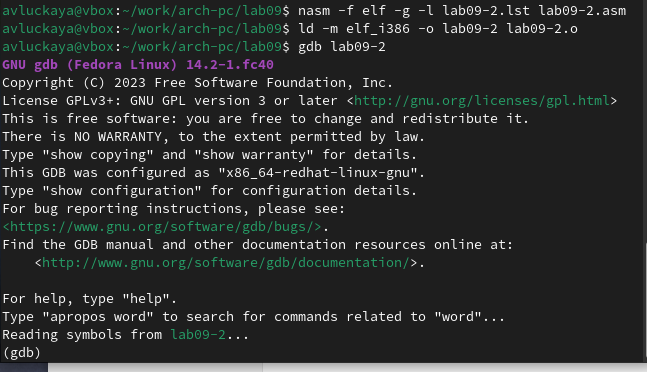


Рис. 7: Загрузка файла

Проверьте работу программы, запустив ее в оболочке GDB с помощью команды run (рис. -fig. 8).

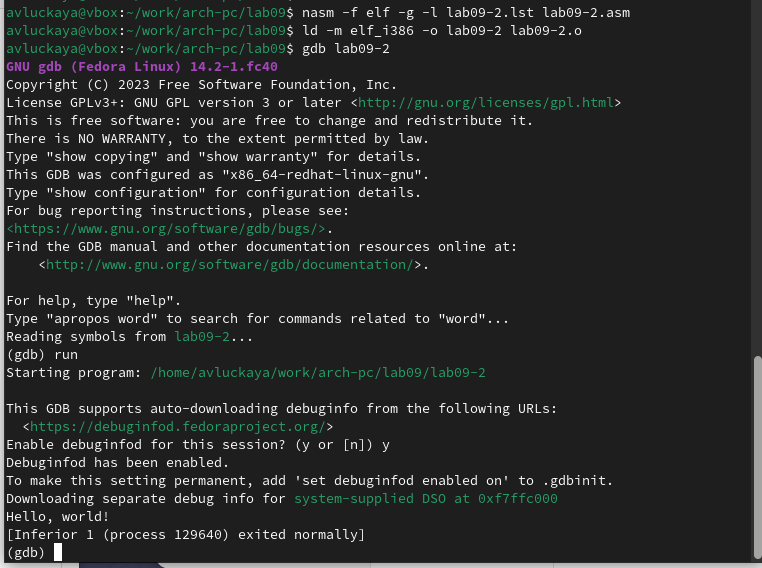


Рис. 8: Запуск программы

Для более подробного анализа программы установливаю брейкпоинт на метку \_start и запускаю её. (рис. -fig. 9).

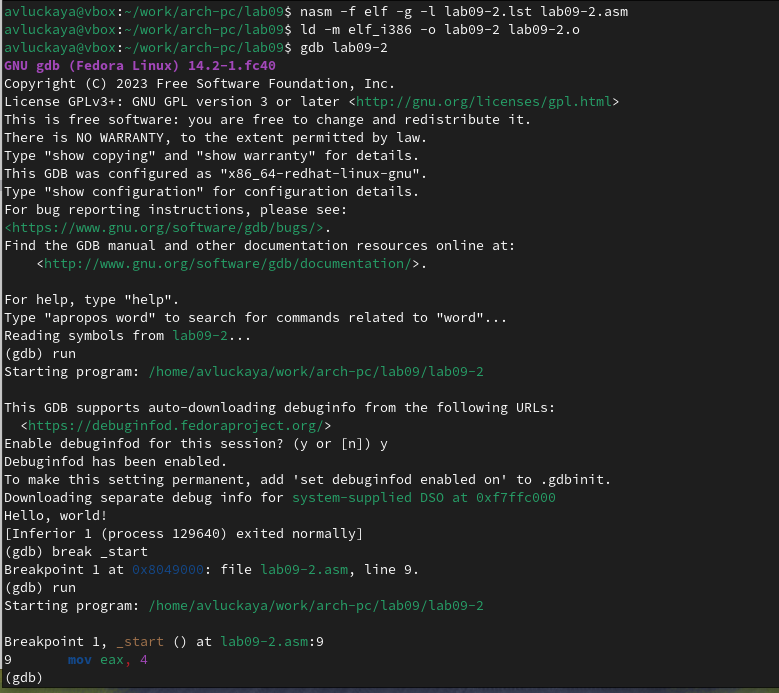


Рис. 9: Запуск отладчика с брейкпоинтом

Просматриваю дисассимилированный код программы с помощью команды disassemble (рис. -fig. 10).

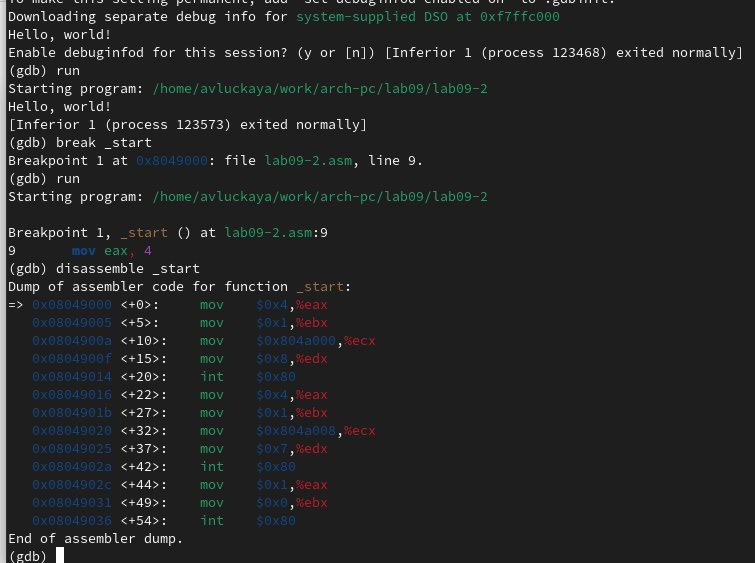


Рис. 10: Просмотр кода

Переключаюсь на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel (рис. -fig. 11).

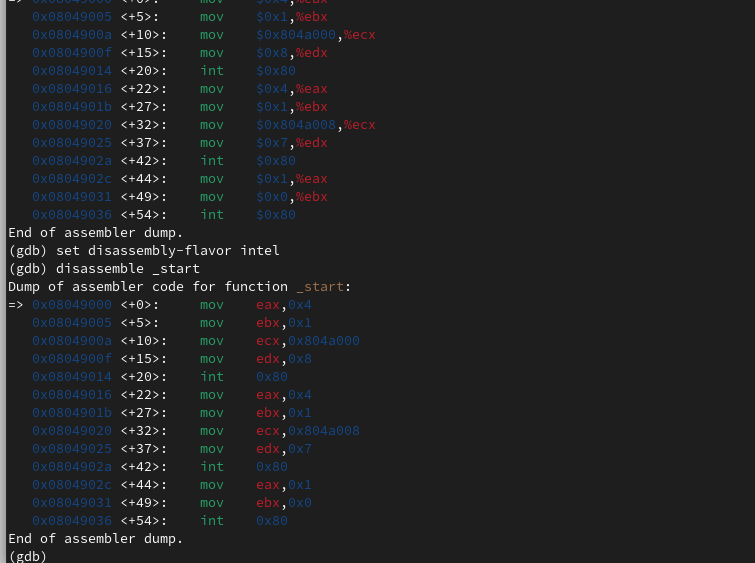


Рис. 11: Переключение

Далее смотрю дисассимилированный код программы, перевожу на команд с синтаксисом Intel (рис. -fig. 12).

Различия между синтаксисом ATT и Intel заключаются в порядке операндов (ATT - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (ATT - pазмер операндов указывается явно с помощью суффиксов, непосредственные операнды предваряются символом $; Intel - Размер операндов неявно определяется контекстом, как ax, eax, непосредственные операнды пишутся напрямую), именах регистров(ATT - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

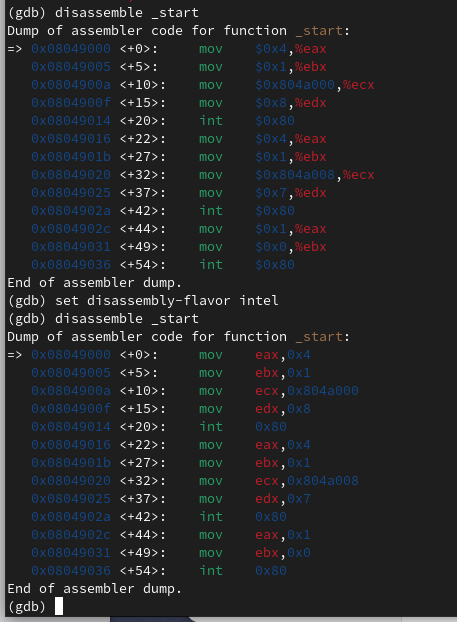


Рис. 12: Дисассимилирование программы

Включаю режим псевдографики для более удобного анализа программы (рис. -fig. 13).

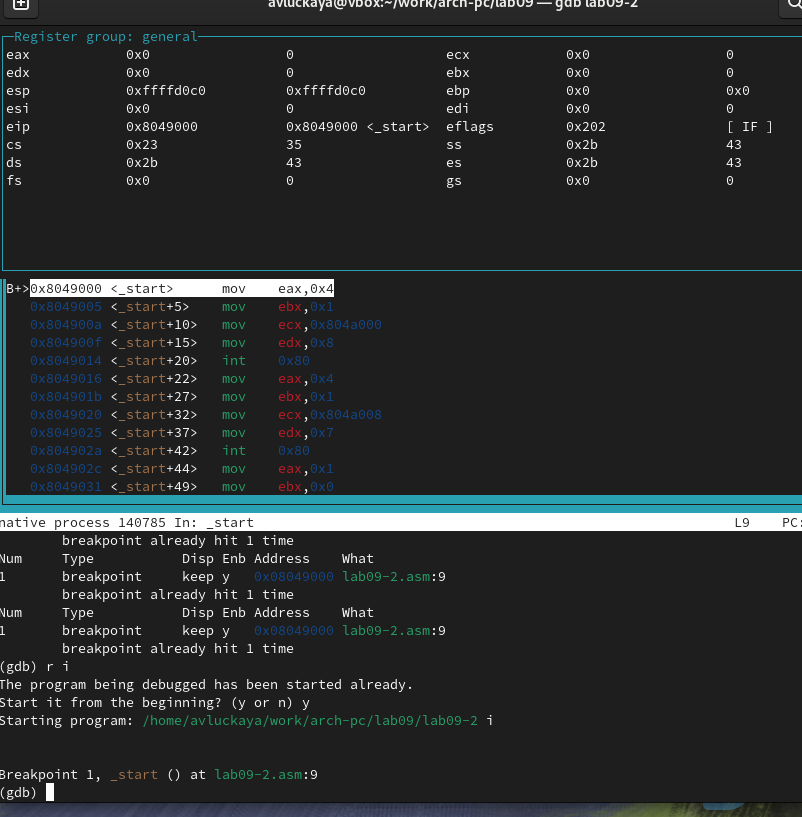


Рис. 13: Режим псевдографики

### 4.1.2 Добавление точек останова

Проверяю в режиме псевдографики, что брейкпоинт сохранился. Устаналиваю еще одну точку останова по адресу инструкции (рис. -fig. 14).

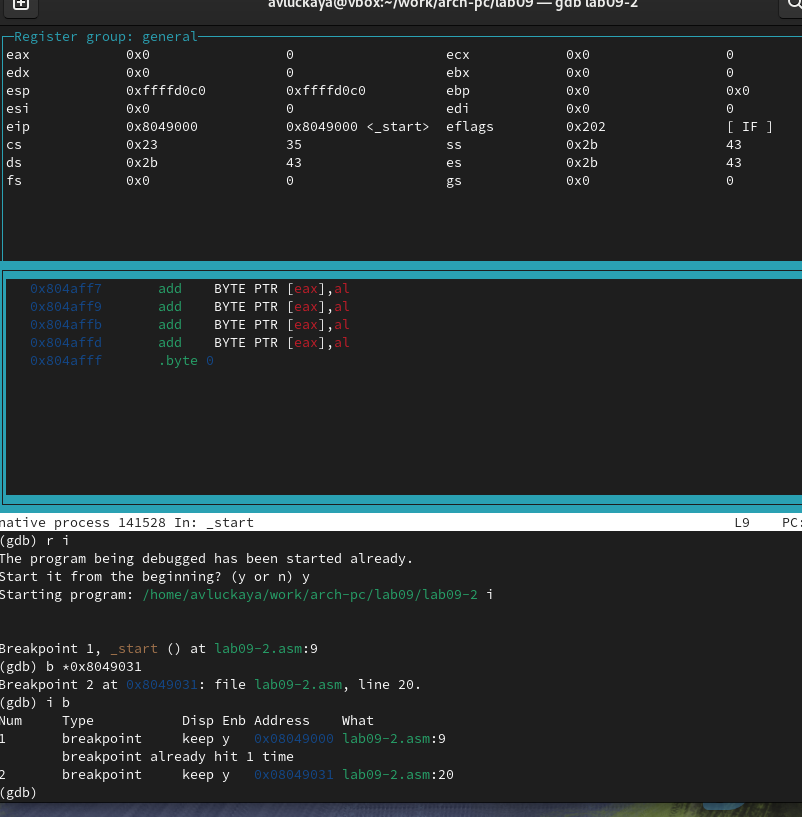


Рис. 14: Добавление второй точки останова

### 4.1.3 Работа с данными программы в GDB

Просматриваю содержимое регистров командой info registers (рис. -fig. 15).

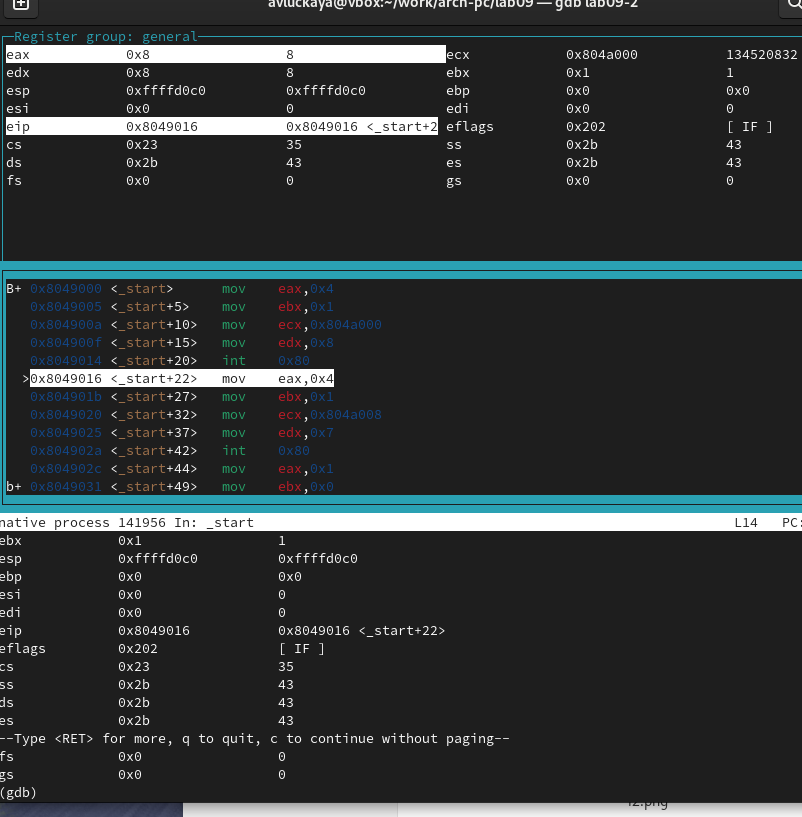


Рис. 15: Просмотр содержимого регистров

Смотрю содержимое переменных по имени и по адресу (рис. -fig. 16).

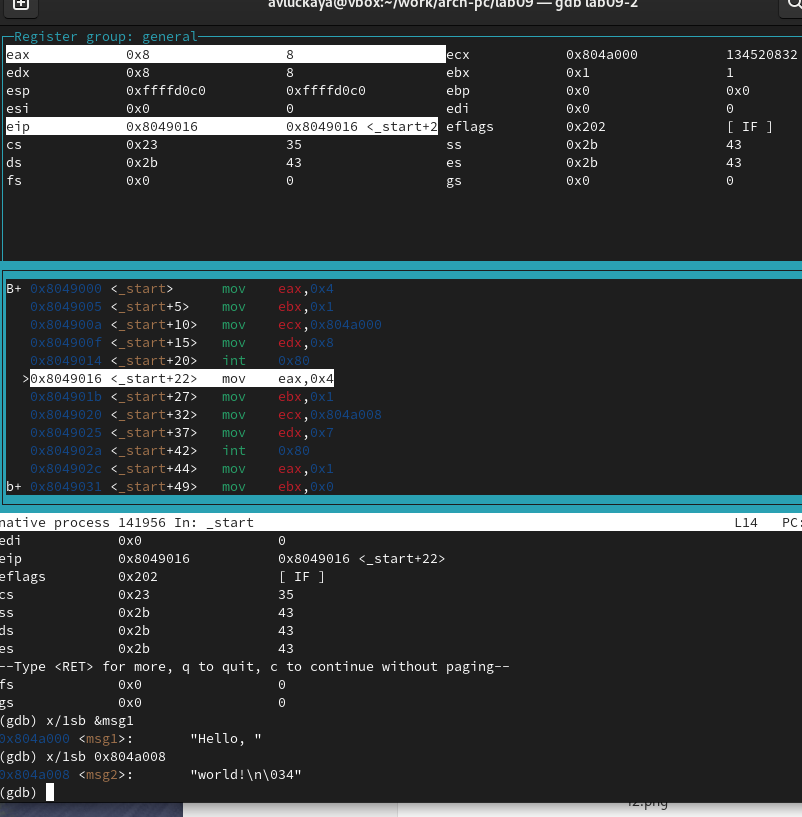


Рис. 16: Просмотр содержимого переменных двумя способами

Меняю содержимое переменных по имени и по адресу (рис. -fig. 17).

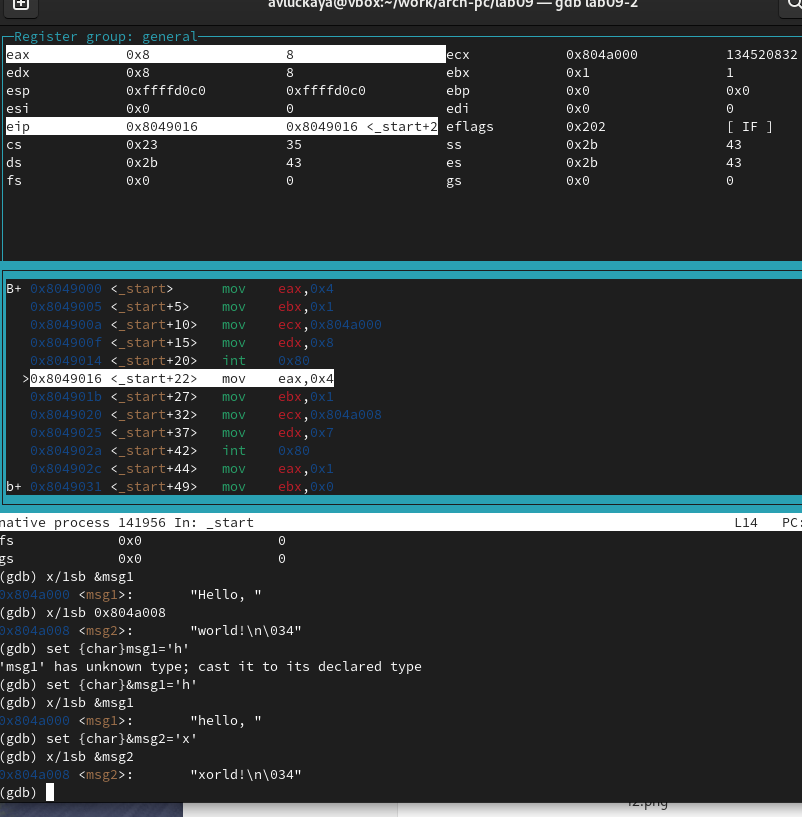


Рис. 17: Изменение содержимого переменных двумя способами

Вывожу в различных форматах значение регистра edx (рис. -fig. 18).

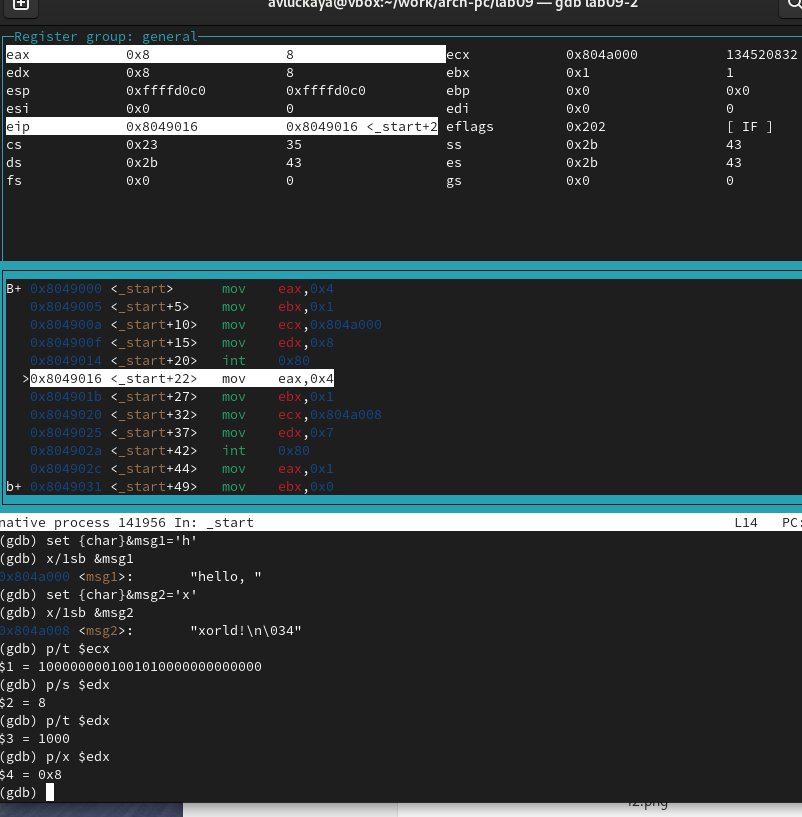


Рис. 18: Просмотр значения регистра разными представлениями

С помощью команды set меняю содержимое регистра ebx (рис. -fig. 19).

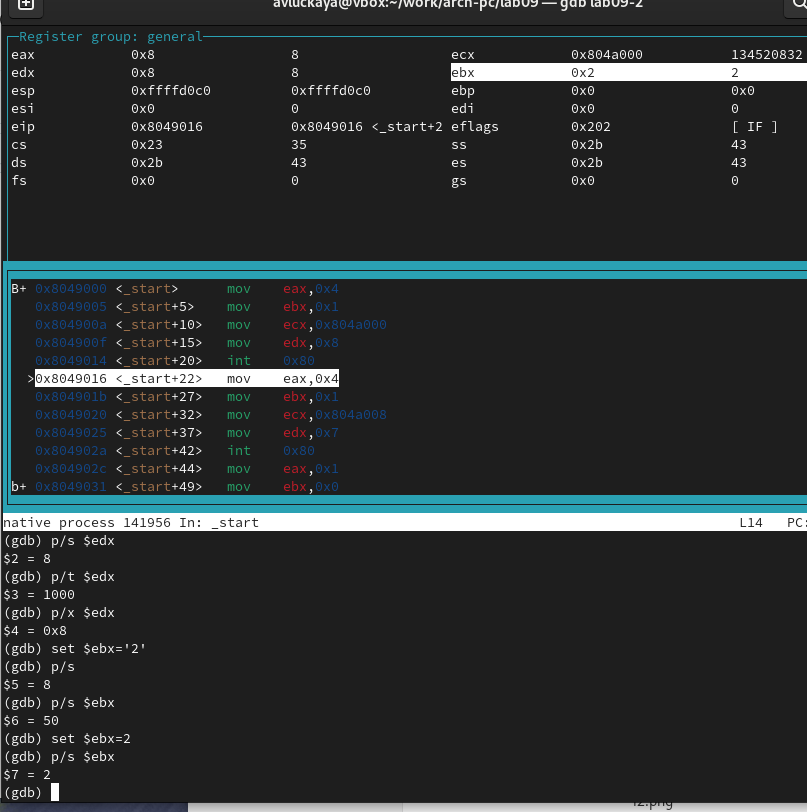


Рис. 19: Примеры использования команды set

### 4.1.4 Обработка аргументов командной строки в GDB

Копирую файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm. Создаю исполняемый файл. Загружаю исполняемый файл в отладчик, указав аргументы. (рис. -fig. 20).

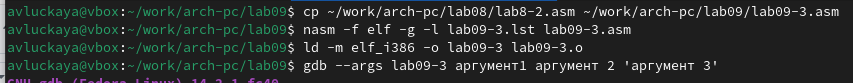


Рис. 20: Подготовка новой программы

Запускаю программу с режиме отладки с указанием аргументов, указываю брейкпопнт и запускаю отладку. Проверяю работу стека, изменяя аргумент команды просмотра регистра esp на +4, число обусловлено разрядностью системы, а указатель void занимает как раз 4 байта, ошибка при аргументе +24 означает, что аргументы на вход программы закончились. (рис. -fig. 16).

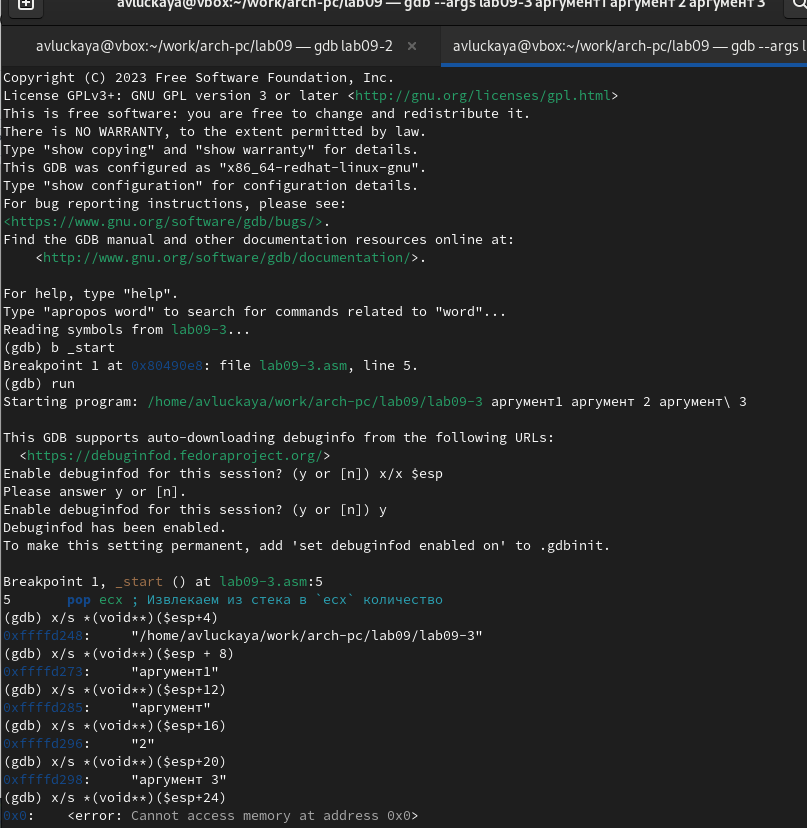


Рис. 21: Проверка работы стека

## 4.2 Задание для самостоятельной работы

1. Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. -fig. 22).

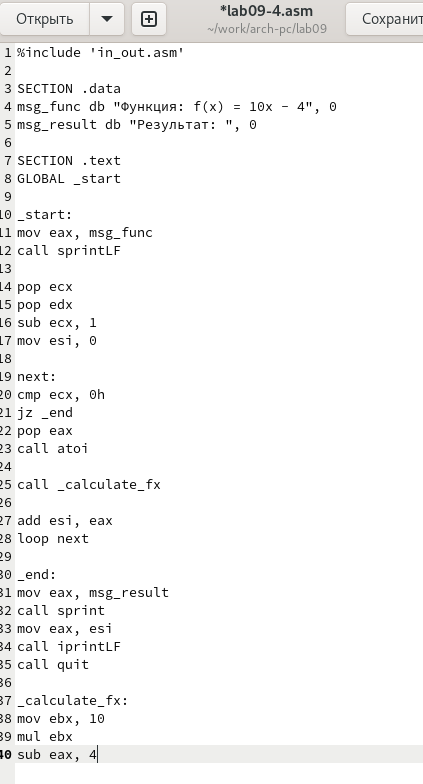


Рис. 22: Измененная программа предыдущей лабораторной работы

1. Запускаю программу в режике отладичка и пошагово через si просматриваю изменение значений регистров через i r. При выполнении инструкции mul ecx можно заметить, что результат умножения записывается в регистр eax, но также меняет и edx. Значение регистра ebx не обновляется напрямую, поэтому результат программа неверно подсчитывает функцию (рис. -fig. 23).

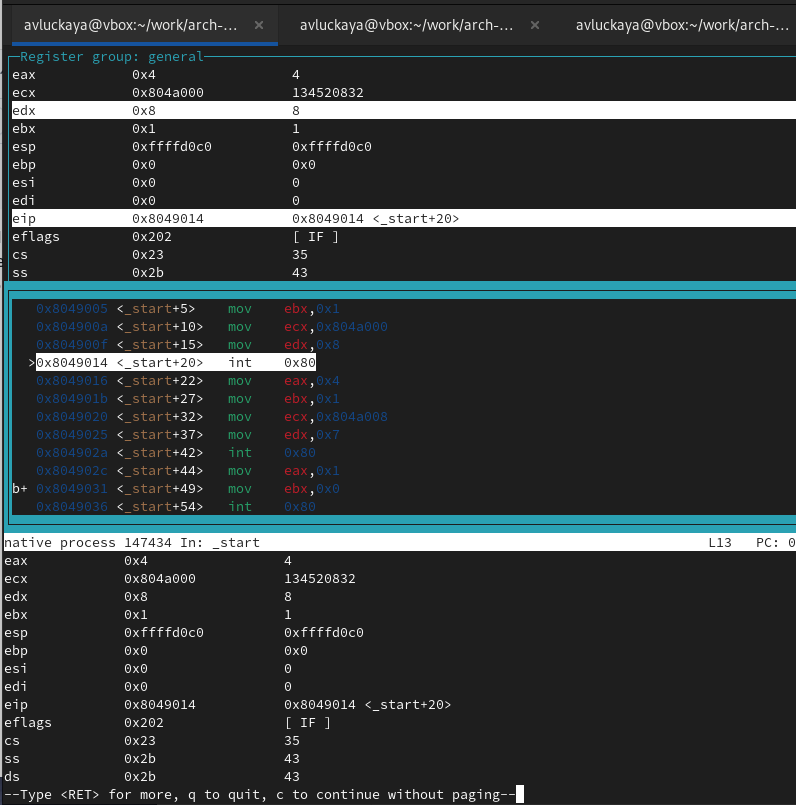


Рис. 23: Поиск ошибки в программе через пошаговую отладку

Исправляю найденную ошибку (рис. -fig. 24).

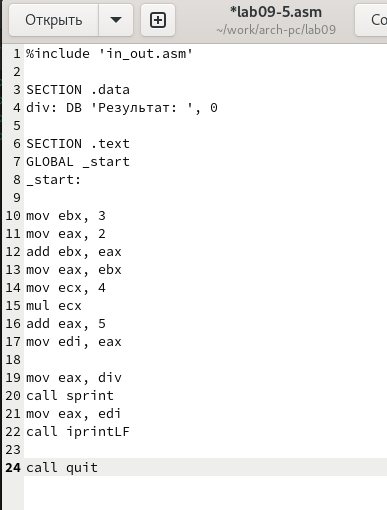


Рис. 24: Исправление ошибки

Проверка выполнения программы (рис. -fig. 25).

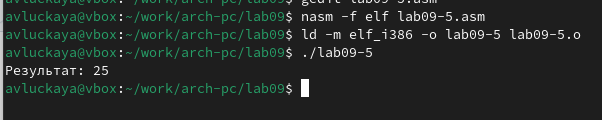


Рис. 25: Проверка корректировок в программме

# 5 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.

# 6 Список литературы

1. https://esystem.rudn.ru/course/view.php?id=112

:::