Отчёт по лабораторной работе №9

Понятие подпрограммы. Отладчик GDB.

Амина Аджигалиева

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

## 2.1 Реализация подпрограмм в NASM

Создаем каталог для программ ЛБ9, и в нем создаем файл (рис. 1).

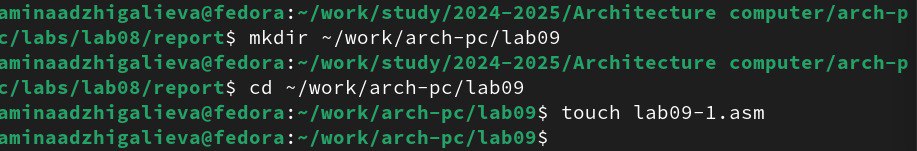


Рис. 1: Новый каталог и файл

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 9.1 (рис. 2).

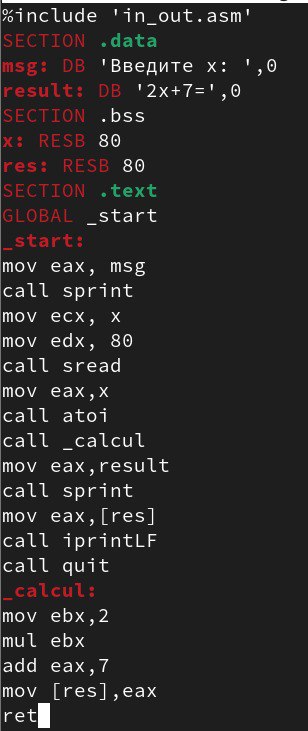


Рис. 2: Код программы

Создаем исполняемый файл и запускаем его (рис. 3).

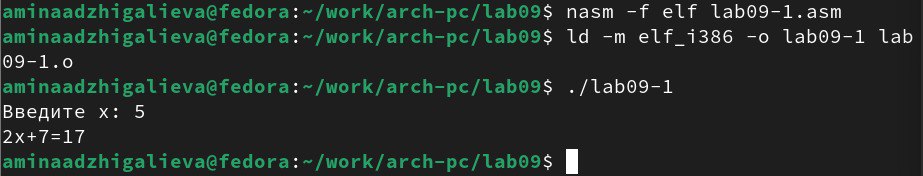


Рис. 3: Запуск программы

Снова открываем файл для редактирования и изменяем его, добавив подпрограмму в подпрограмму (рис. 4).

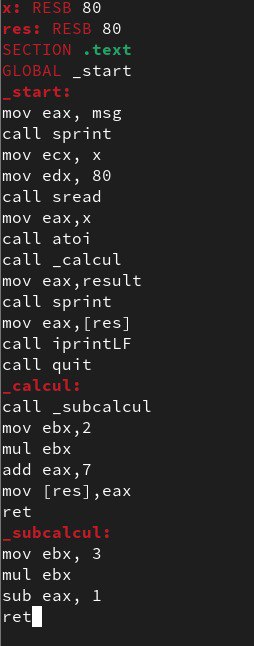


Рис. 4: Подпрограмма

Создаем исполняемый файл и запускаем его (рис. 5).



Рис. 5: Запуск программы

## 2.2 Отладка программам с помощью GDB

Создаем новый файл в каталоге (рис. 6).

Новый файл

Рис. 6: Новый файл

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 9.2 (рис. 7).

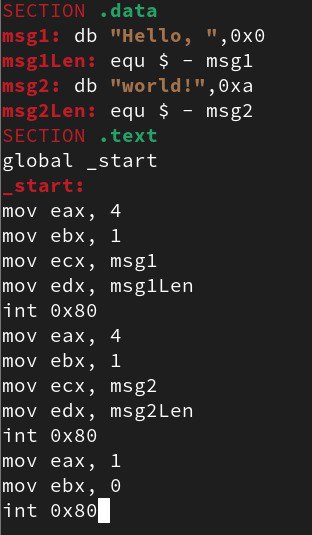


Рис. 7: Код программы

Получаем исходный файл с использованием отладчика gdb (рис. 8).

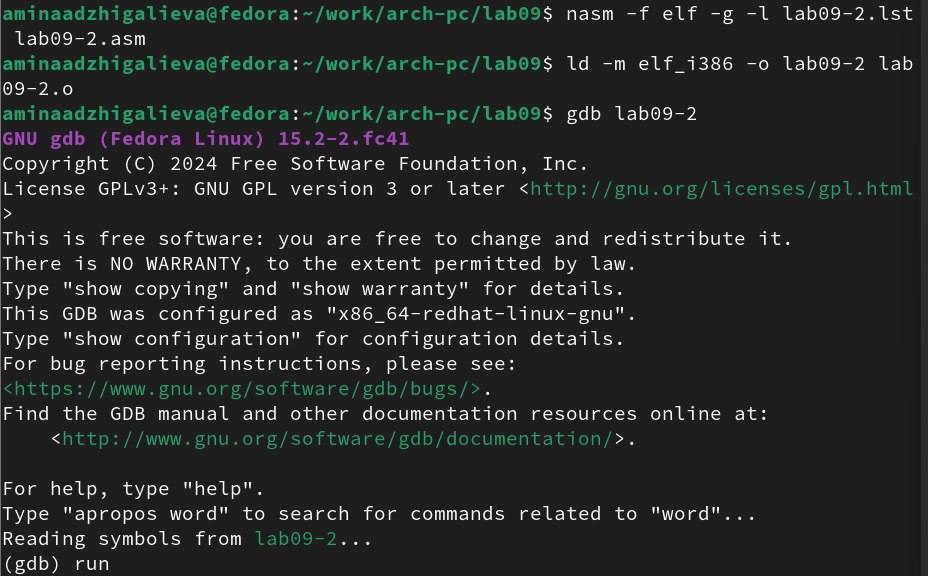


Рис. 8: Исходный файл

Загрузим исполняемый файл в отладчик (рис. 9).



Рис. 9: Загрузка

Проверим работу программы, запустив ее в оболочке GDB с помощью команды run (рис. 10).

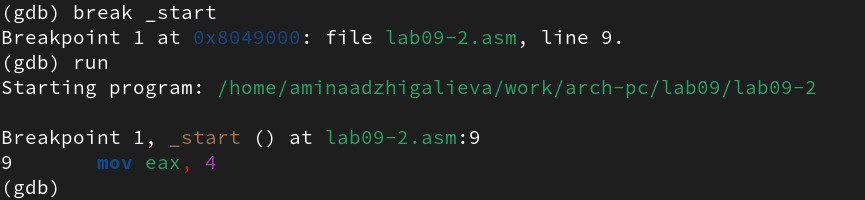


Рис. 10: Проверка программы

Устанавливаем брейкпоинт на метку \_start и запускаем программу (рис. 11).



Рис. 11: Брейкпоинт

Посмотрим дисассимилированный код программы (рис. 12).

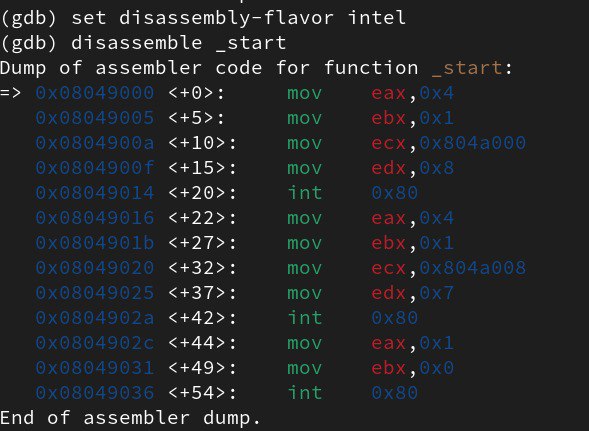


Рис. 12: Дисассимилированный код

Переключимся на отображение команд с Intel’овским синтаксисом (рис. 13).

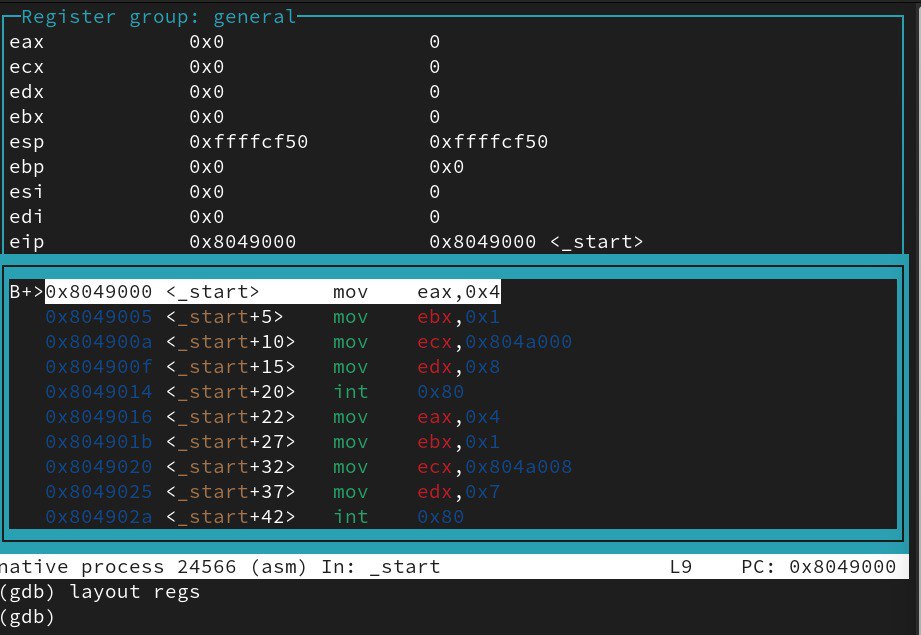


Рис. 13: Intel’овский синтаксис

Различия отображения синтаксиса машинных команд в режимах ATT и Intel:

1.Порядок операндов: В ATT синтаксисе порядок операндов обратный, сначала указывается исходный операнд, а затем - результирующий операнд. В Intel синтаксисе порядок обычно прямой, результирующий операнд указывается первым, а исходный - вторым.

2.Разделители: В ATT синтаксисе разделители операндов - запятые. В Intel синтаксисе разделители могут быть запятые или косые черты (/).

3.Префиксы размера операндов: В ATT синтаксисе размер операнда указывает ся перед операндом с использованием префиксов, таких как “b” (byte), “w” (word), “l” (long) и “q” (quadword). В Intel синтаксисе размер операнда указывается после операнда с использованием суффиксов, таких как “b”, “w”, “d” и “q”.

4.Знак операндов: В ATT синтаксисе операнды с позитивными значениями предваряются символом ““.𝐼𝑛𝑡𝑒𝑙””.

5.Обозначение адресов: В ATT синтаксисе адреса указываются в круглых скобках. В Intel синтаксисе адреса указываются без скобок.

6.Обозначение регистров: В ATT синтаксисе обозначение регистра начинается с символа “%”. В Intel синтаксисе обозначение регистра может начинаться с символа “R” или “E” (например, “%eax” или “RAX”).

Включим режим псевдографики для более удобного анализа программы (рис. 14).

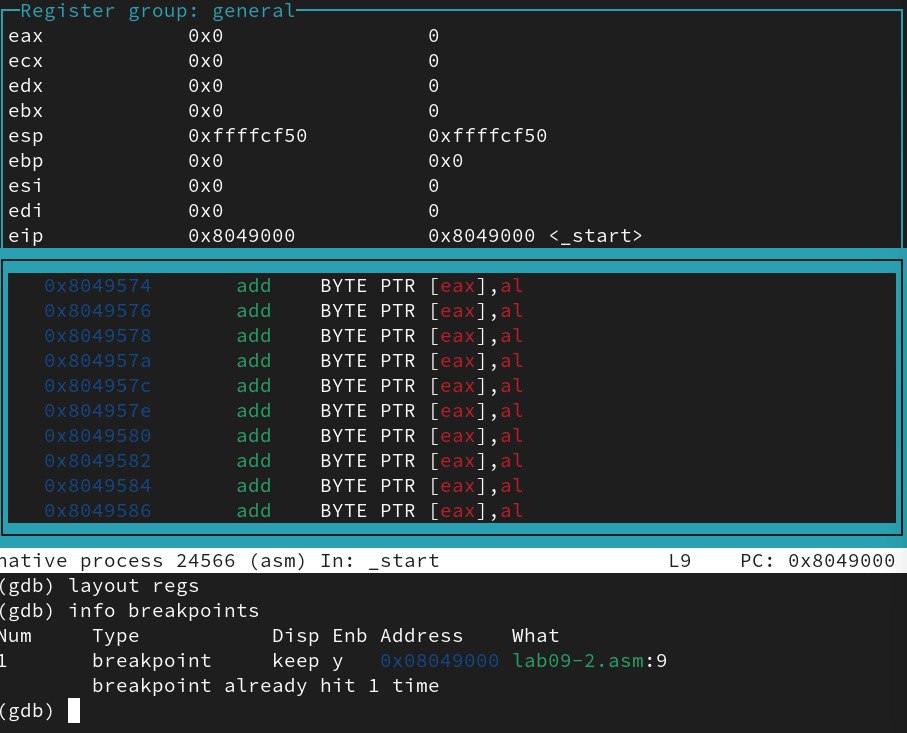


Рис. 14: Режим псевдографики

## 2.3 Добавление точек останова

Проверим это с помощью команды info breakpoints (рис. 15).

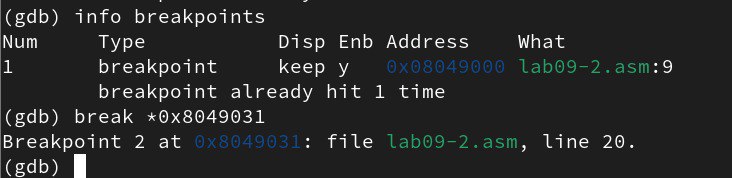


Рис. 15: Info breakpoints

Посмотрим информацию о всех установленных точках останова (рис. 16).

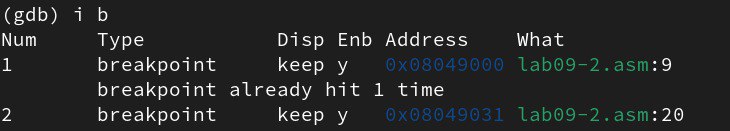


Рис. 16: Точки останова

## 2.4 Работа с данными программы в GDB

Выполняем 5 инструкций командой si (рис. 17).

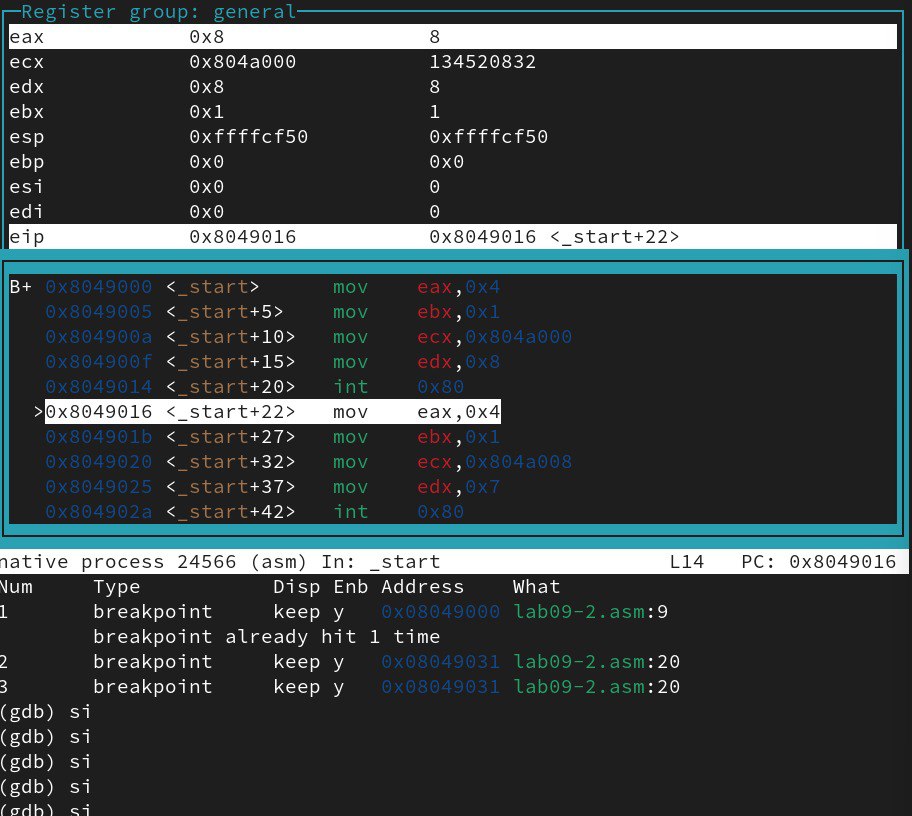


Рис. 17: Si

Во время выполнения команд менялись регистры: ebx, ecx, edx,eax, eip.

Смотрим значение переменной msg1 по имени (рис. 18).

msg1

Рис. 18: msg1

Смотрим значение переменной msg2 по адресу (рис. 19).



Рис. 19: msg2

Изменим первый символ переменной msg1 (рис. 20).

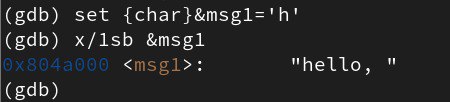


Рис. 20: Замена символа

Изменим первый символ переменной msg2 (рис. 21).

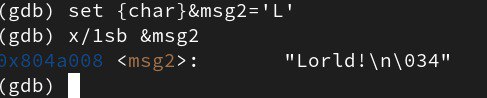


Рис. 21: Замена символа

Смотрим значение регистра edx в разных форматах (рис. 22).

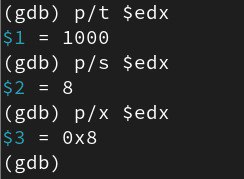


Рис. 22: Значение регистра

Изменяем регистор ebx (рис. 23).

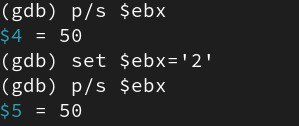


Рис. 23: Замена регистра

Выводятся разные значения, так как команда без кавычек присваивает регистру вводимое значение.

Прописываем команды для завершения программы и выхода из GDB (рис. 24).



Рис. 24: Завершение программы

## 2.5 Обработка аргументов командной строки в GDB

Копируем файл lab8-2.asm в файл с именем lab09-3.asm (рис. 25).



Рис. 25: Копия файла

Создаем исполняемый файл и запускаем его в отладчике GDB (рис. 26).

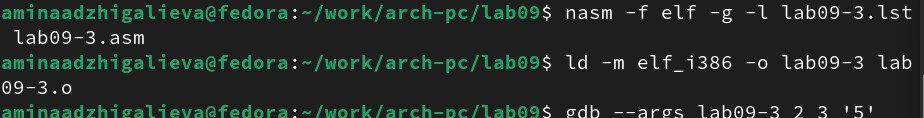


Рис. 26: Запуск программы

Установим точку останова перед первой инструкцией в программе и запустим ее (рис. 27).



Рис. 27: Точка останова

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (рис. 28).

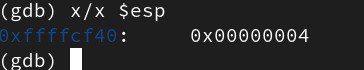


Рис. 28: Адрес вершины стека

Смотрим позиции стека по разным адресам (рис. 29).

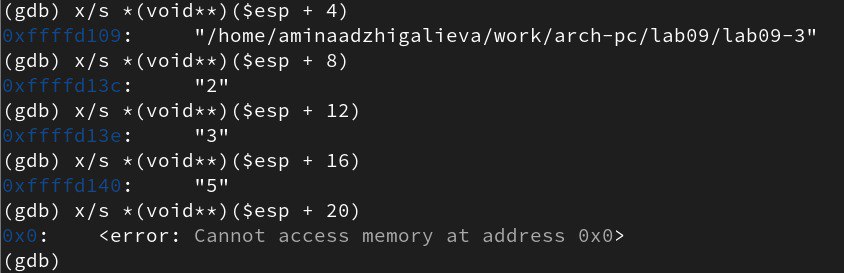


Рис. 29: Позиции стека

Шаг изменения адреса равен 4 потому что адресные регистры имеют размерность 32 бита(4 байта).

## 2.6 Задание для самостоятельной работы

Задание 1. Копируем файл lab8-4.asm в файл с именем lab09-3.asm (рис. 30).



Рис. 30: Копия файла

Меняем код, создавая подпрограмму (рис. 31).

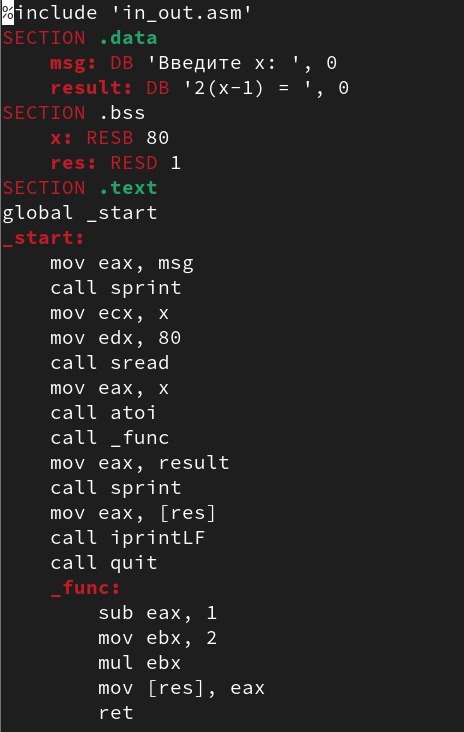


Рис. 31: Код программы

Создаем исполняемый файл и запускаем его (рис. 32).

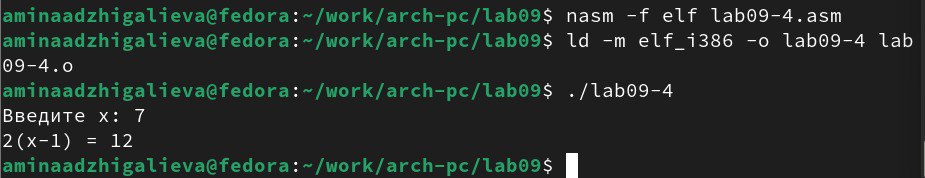


Рис. 32: Запуск программы

Задание 2. Создаем новый файл (рис. 33).

Новый файл

Рис. 33: Новый файл

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 9.3 (рис. 34).

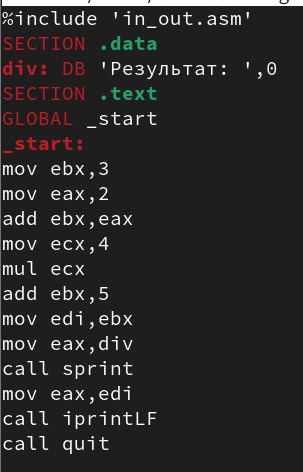


Рис. 34: Код программы

Создаем исполняемый файл и запускаем его (рис. 35).

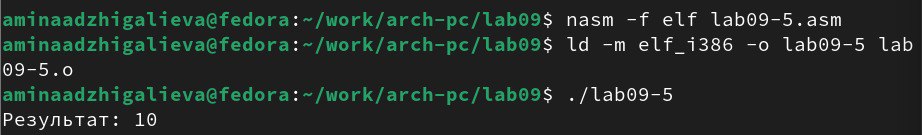


Рис. 35: Запуск программы

Создаем исполняемый файл и запускаем его в отладчике GDB и смотрим на изменение регистров командой si (рис. 36).

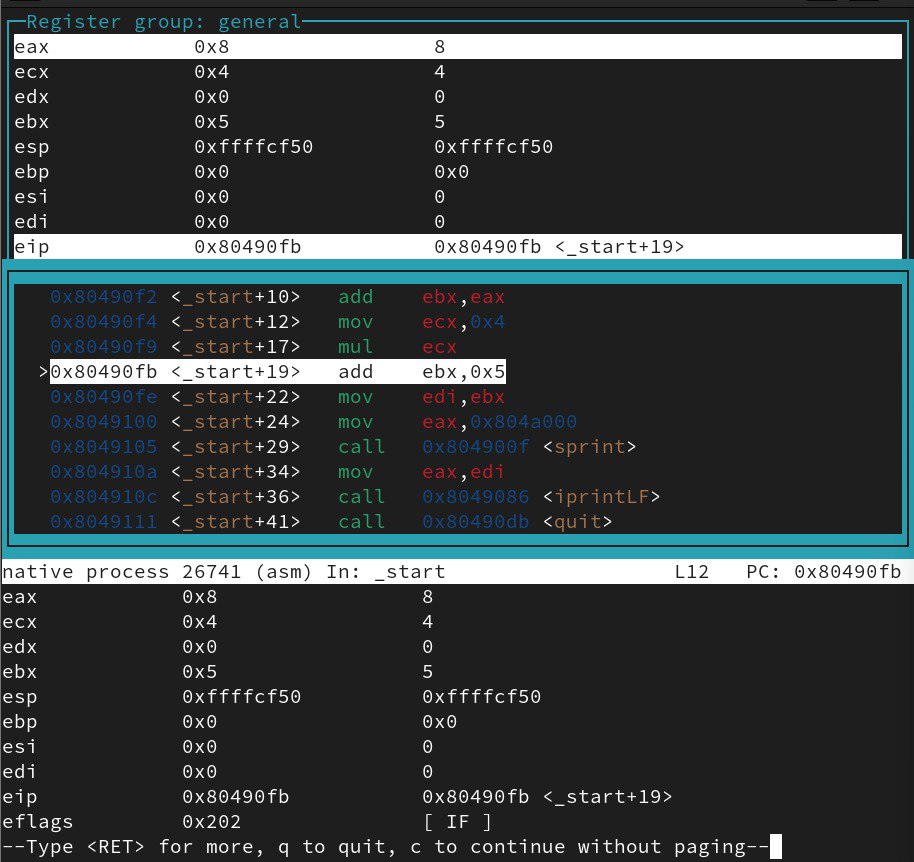


Рис. 36: Отладчик GDB

Изменяем программу для корректной работы (рис. 37).

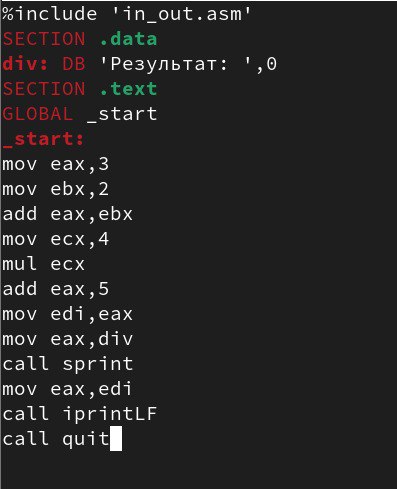


Рис. 37: Замена программы

Создаем исполняемый файл и запускаем его (рис. 38).

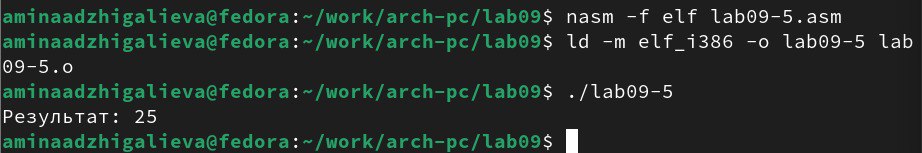


Рис. 38: Запуск программы

# 3 Выводы

Мы познакомились с методами отладки при помощи GDB и его возможностями.