Отчёт по лабораторной работе 9

Дисциплина: архитектура компьютера

Айдарбекова Алия НММбд-01-23

Содержание

# 1 Цель работы

Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

# 2 Задание

1. Изучение подпрограмм в ассемблере
2. Изучение функций отладчика GDB
3. Изучение передачи аргументов с помощью отладчика
4. Выполнение заданий, рассмотрение примеров
5. Выполнение заданий 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.

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

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.

Подпрограмма — это, как правило, функционально законченный участок кода, который можно многократно вызывать из разных мест программы. В отличие от простых переходов из подпрограмм существует возврат на команду, следующую за вызовом. Если в программе встречается одинаковый участок кода, его можно оформить в виде подпрограммы, а во всех нужных местах поставить её вызов. При этом подпрограмма будет содержаться в коде в одном экземпляре, что позволит уменьшить размер кода всей программы.

Для вызова подпрограммы из основной программы используется инструкция call, которая заносит адрес следующей инструкции в стек и загружает в регистр eip адрес соответствующей подпрограммы, осуществляя таким образом переход. Затем начинается выполнение подпрограммы, которая, в свою очередь, также может содержать подпрограммы.

Подпрограмма завершается инструкцией ret, которая извлекает из стека адрес, занесённый туда соответствующей инструкцией call, и заносит его в eip. После этого выполнение основной программы возобновится с инструкции, следующей за инструкцией call.

# 4 Выполнение лабораторной работы

## 4.1 Реализация подпрограмм в NASM

Для начала я создала новую директорию и перешла в нее, чтобы выполнить лабораторную работу номер 9. Затем я создала файл с именем lab9-1.asm.

В качестве примера рассмотрим программу, которая вычисляет арифметическое выражение f(x) = 2x + 7 с использованием подпрограммы calcul. В данном примере значение переменной x вводится с клавиатуры, а само выражение вычисляется внутри подпрограммы. (рис. [[1](#fig:001)]) (рис. [[2](#fig:002)])

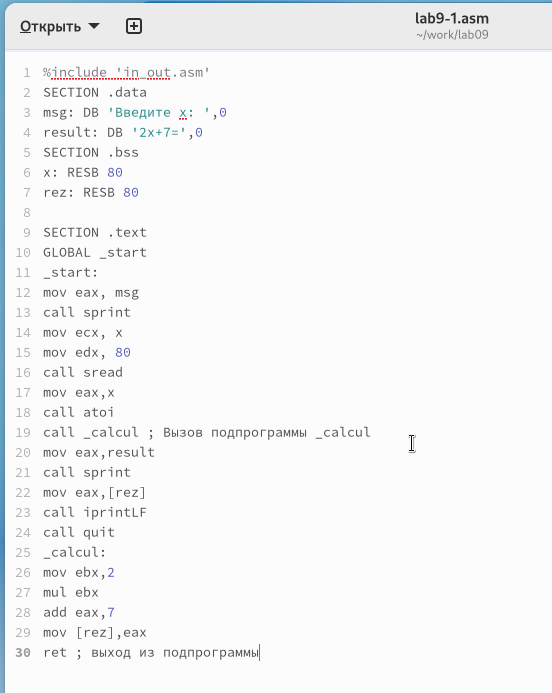


Figure 1: Изменение кода lab9-1.asm

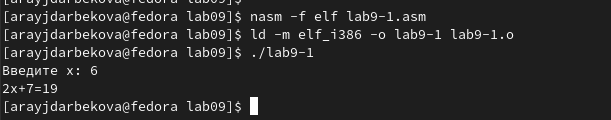


Figure 2: Компиляция текста программы lab9-1.asm

После этого я внесла изменения в текст программы, добавив подпрограмму subcalcul внутрь подпрограммы calcul. Это позволяет вычислить составное выражение f(g(x)), где значение x также вводится с клавиатуры. Функции определены следующим образом: f(x) = 2x + 7, g(x) = 3x - 1. (рис. [[3](#fig:003)]) (рис. [[4](#fig:004)])

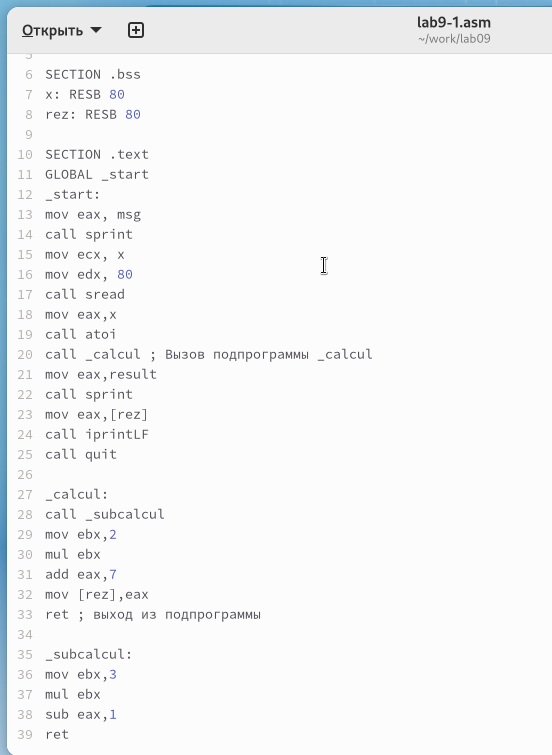


Figure 3: Изменение кода lab9-1.asm

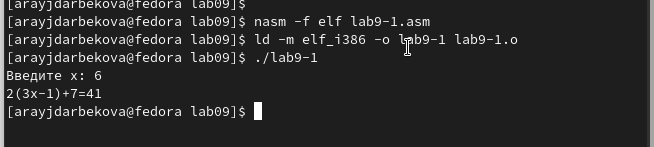


Figure 4: Компиляция текста программы lab9-1.asm

## 4.2 Отладка программам с помощью GDB

Я создала файл с именем lab9-2.asm и внесла в него текст программы из Листинга 9.2. Эта программа предназначена для вывода сообщения “Hello world!”. (рис. [[5](#fig:005)])

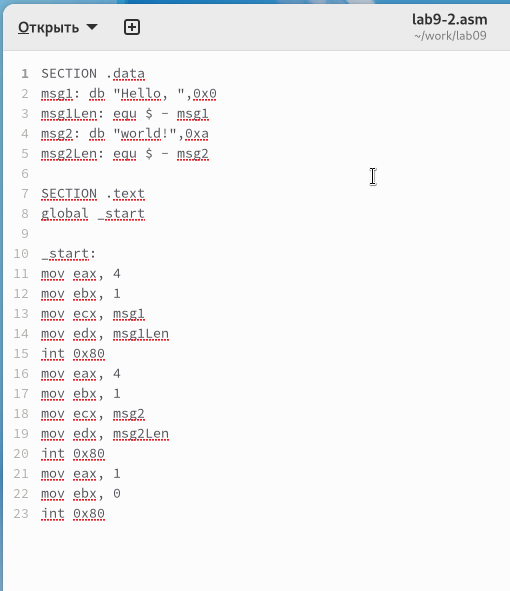


Figure 5: Изменение кода lab9-2.asm

Далее я скомпилировала файл и получила исполняемый файл. Чтобы добавить отладочную информацию для работы с отладчиком GDB, использовала ключ “-g”.

Затем загрузила полученный исполняемый файл в отладчик GDB и проверила его работу, запустив программу с помощью команды “run” или “r”.(рис. [[6](#fig:006)])

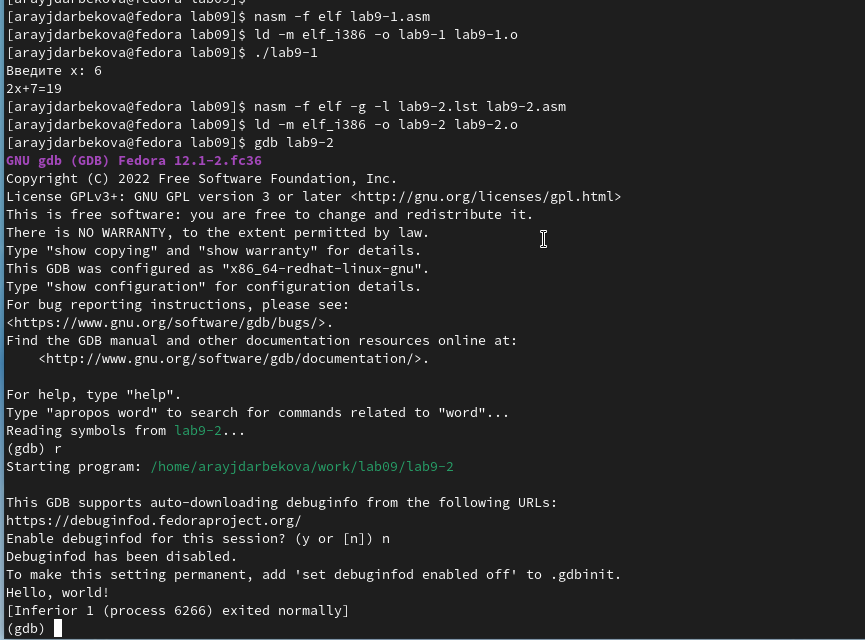


Figure 6: Компиляция текста программы lab9-2.asm в отладчике

Для более детального анализа программы я установила точку остановки на метке “start”, с которой начинается выполнение любой ассемблерной программы, и запустила ее. Затем просмотрела дизассемблированный код программы. (рис. [[7](#fig:007)]) (рис. [[8](#fig:008)])

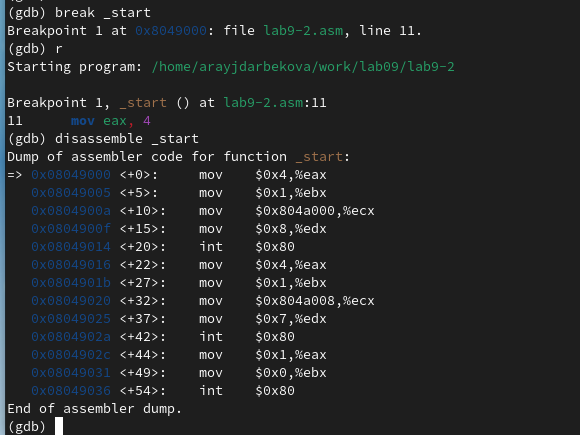


Figure 7: Дизассемблированный код

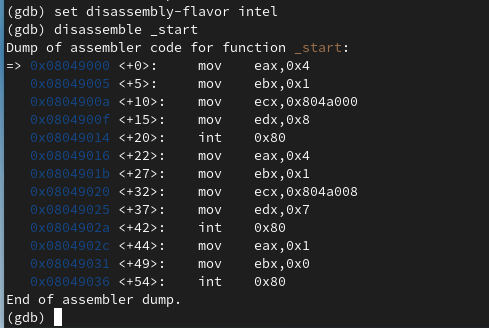


Figure 8: Дизассемблированный код в режиме интел

Чтобы проверить точку остановки по имени метки “\_start”, я использовала команду “info breakpoints” или “i b”. Затем установила еще одну точку остановки по адресу инструкции, определив адрес предпоследней инструкции “mov ebx, 0x0” (рис. [[9](#fig:009)])

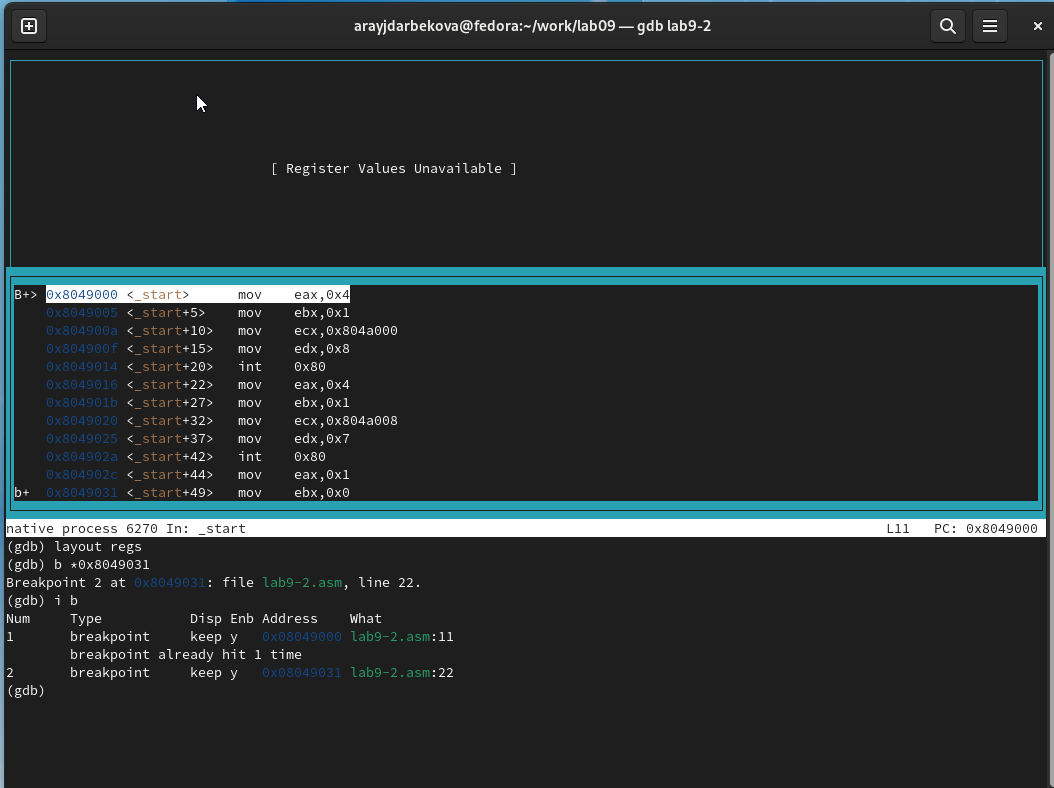


Figure 9: Точка остановки

В отладчике GDB я могу просматривать содержимое ячеек памяти и регистров, а также изменять значения регистров и переменных. Я выполнила 5 инструкций с помощью команды ‘stepi’ (сокращенно ‘si’) и отследила изменение значений регистров. (рис. [[10](#fig:010)]) (рис. [[11](#fig:011)])

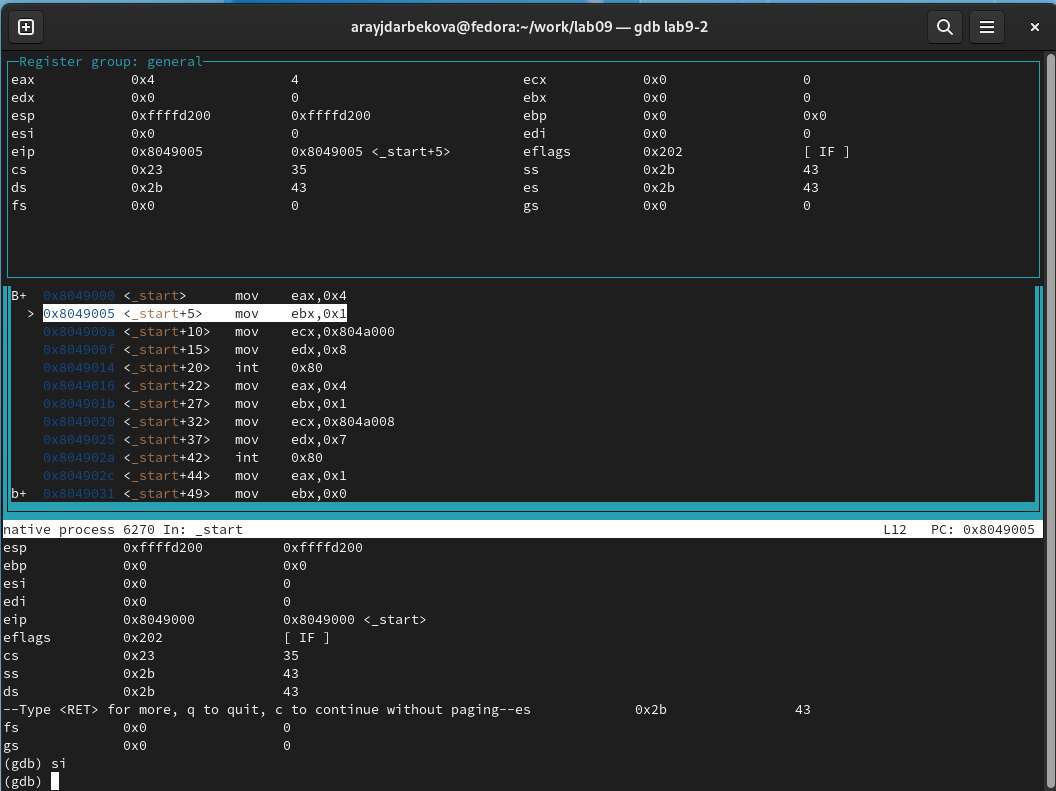


Figure 10: Изменение регистров

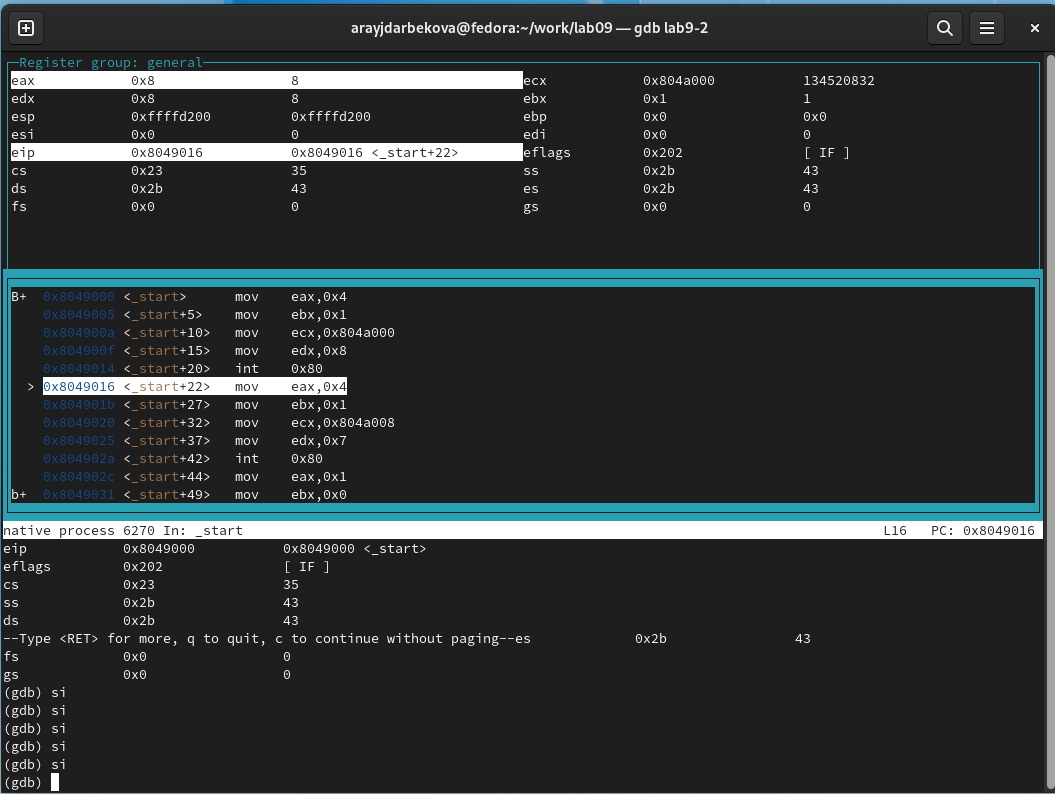


Figure 11: Изменение регистров

Для просмотра значения переменной msg1 по имени и получения нужных данных, использовала соответствующую команду.

Для изменения значения регистра или ячейки памяти, использовала команду set, указав имя регистра или адрес в качестве аргумента. (рис. [[12](#fig:012)])

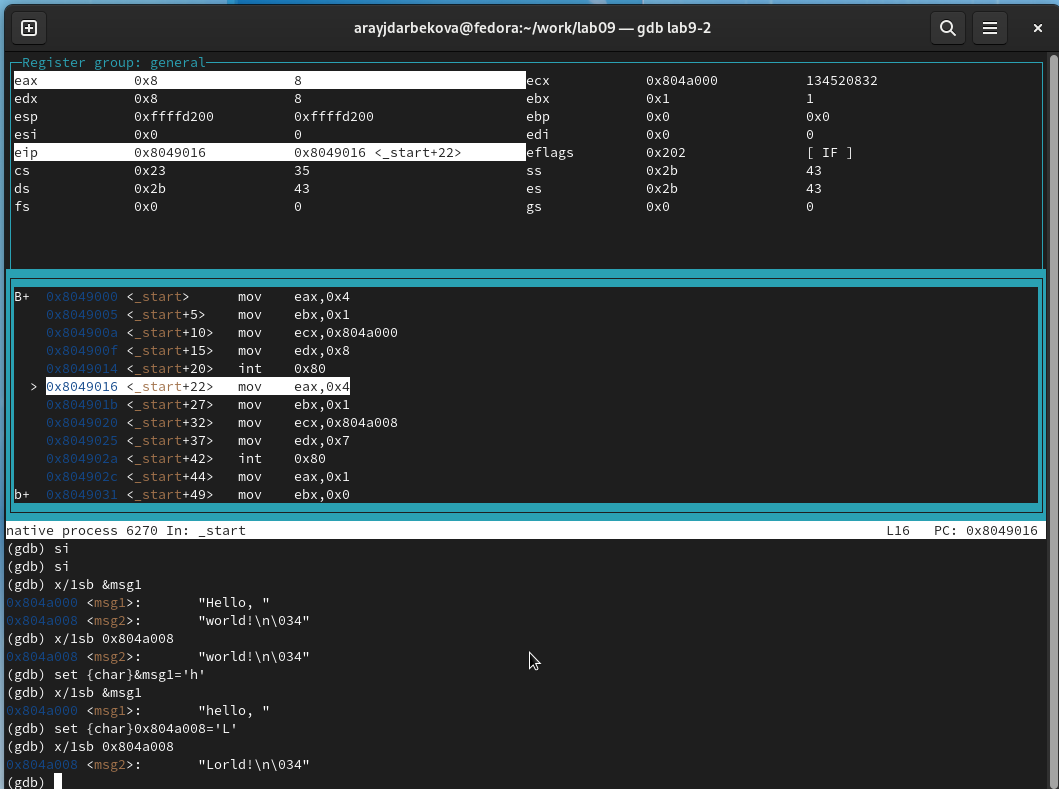


Figure 12: Изменение значения переменной

Я успешно изменила первый символ переменной msg1.(рис. [[13](#fig:013)])

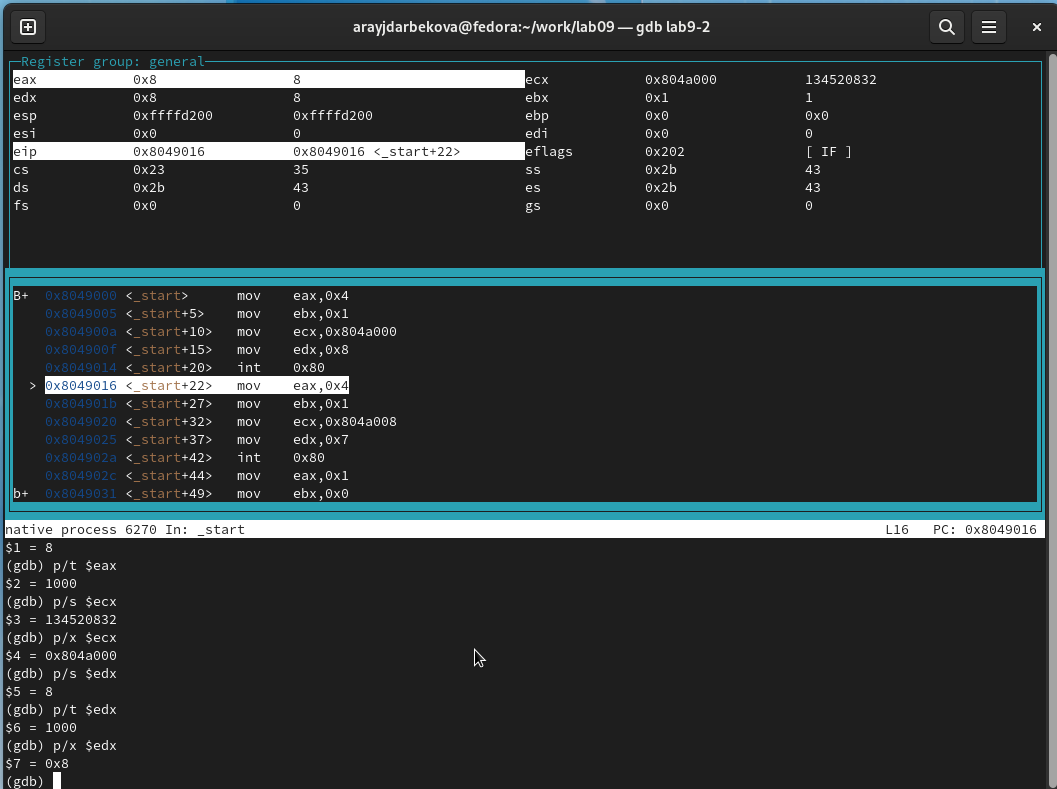


Figure 13: Вывод значения регистра

Также, с помощью команды set, я изменила значение регистра ebx на нужное значение. (рис. [[14](#fig:014)])

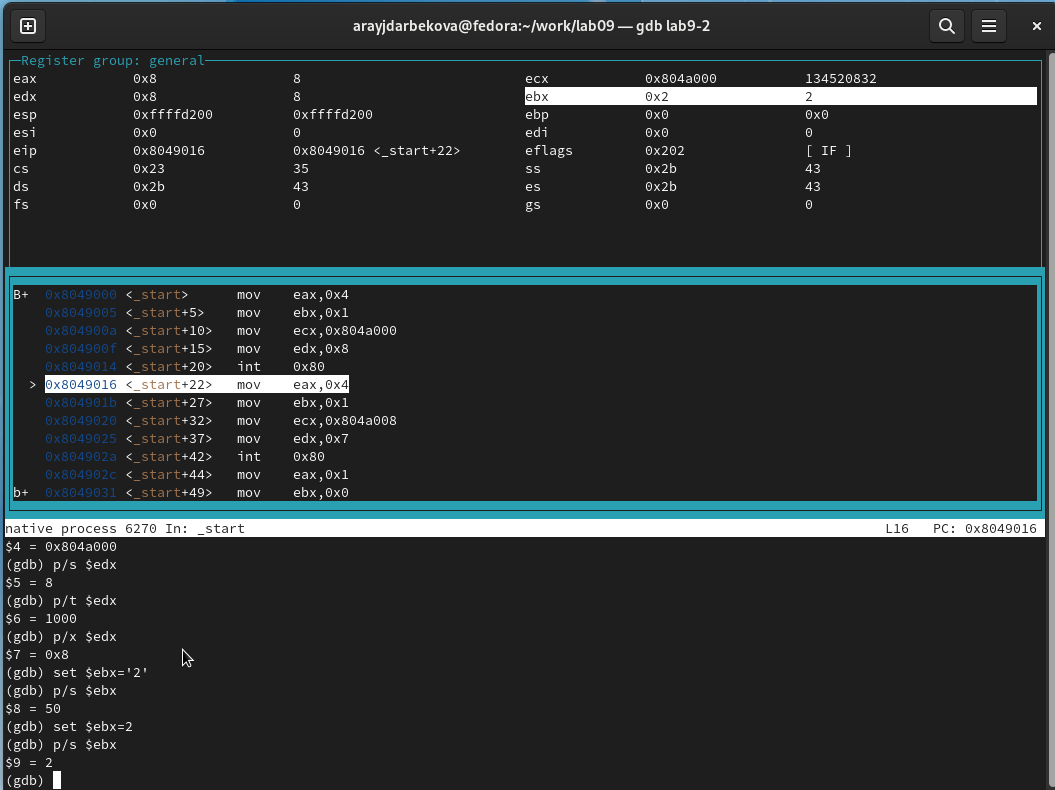


Figure 14: Вывод значения регистра

Я скопировала файл lab8-2.asm, который был создан во время выполнения лабораторной работы №8. Этот файл содержит программу для вывода аргументов командной строки. Затем создала исполняемый файл из скопированного файла.

Для загрузки программы с аргументами в gdb использовала ключ –args и загрузила исполняемый файл в отладчик с указанными аргументами. Я установила точку останова перед первой инструкцией программы и запустила ее.

Адрес вершины стека, где хранится количество аргументов командной строки (включая имя программы), хранится в регистре esp. По этому адресу я нашла число, указывающее количество аргументов. В данном случае увидела, что количество аргументов равно 5, включая имя программы lab9-3 и сами аргументы: аргумент1, аргумент2 и ‘аргумент 3’.

Я также просмотрела остальные позиции стека. По адресу [esp+4] находится адрес в памяти, где располагается имя программы. По адресу [esp+8] хранится адрес первого аргумента, по адресу [esp+12] - второго и так далее. Шаг изменения адреса равен 4, так как каждый следующий адрес на стеке находится на расстоянии 4 байт от предыдущего ([esp+4], [esp+8], [esp+12]). (рис. [[15](#fig:015)])

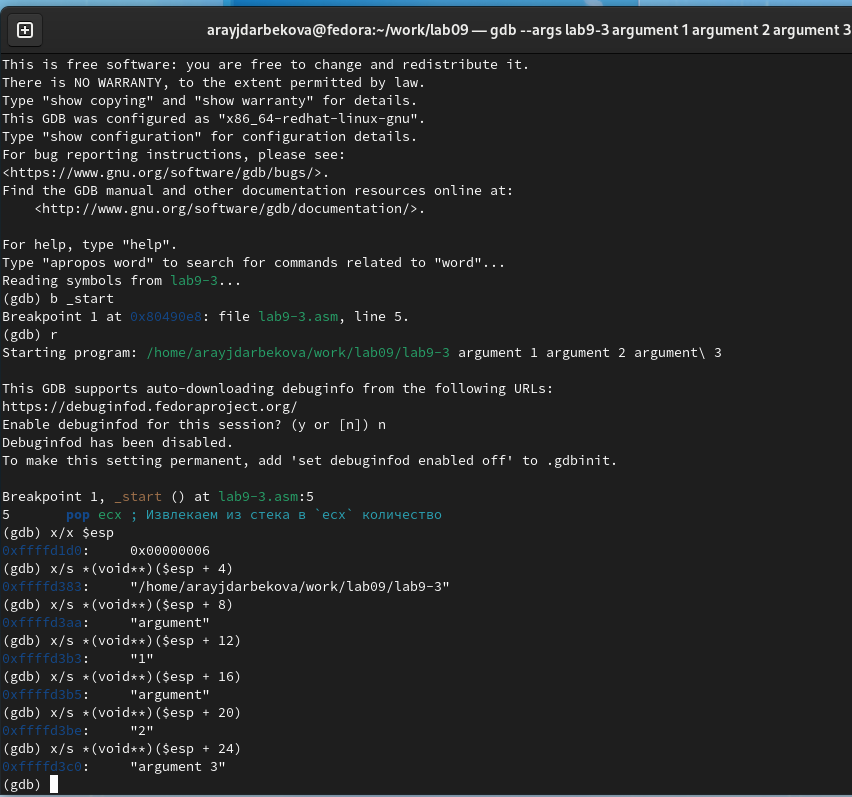


Figure 15: Вывод значения регистра

## 4.3 Задание для самостоятельной работы

Я переписала программу из лабораторной работы №8, задание №1, чтобы реализовать вычисление значения функции f(x) как подпрограмму. (рис. [[16](#fig:016)]) (рис. [[17](#fig:017)])

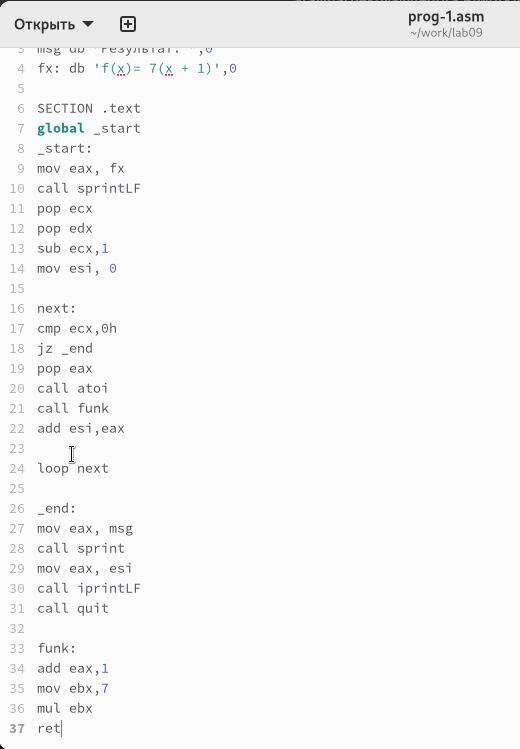


Figure 16: Изменение кода prog-1.asm

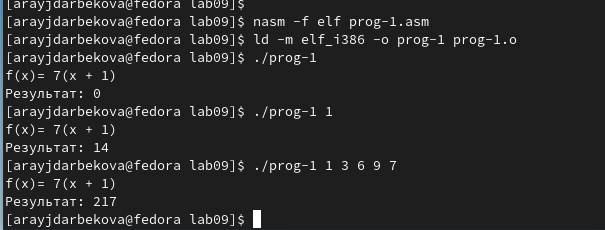


Figure 17: Компиляция текста программы prog-1.asm

Приведенный ниже код представляет программу для вычисления выражения . Однако, при запуске программа дает неверный результат.

Я провела анализ изменений значений регистров с помощью отладчика GDB и обнаружила ошибку: перепутан порядок аргументов у инструкции add. Кроме того, заметила, что по окончании работы в регистр edi передается значение ebx вместо eax.(рис. [[18](#fig:018)]) (рис. [[19](#fig:019)])

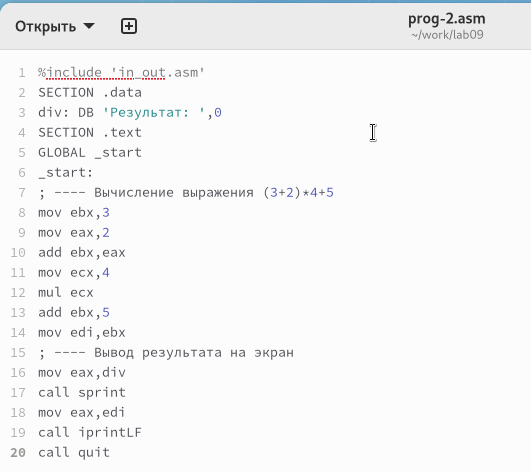


Figure 18: Код с ошибкой

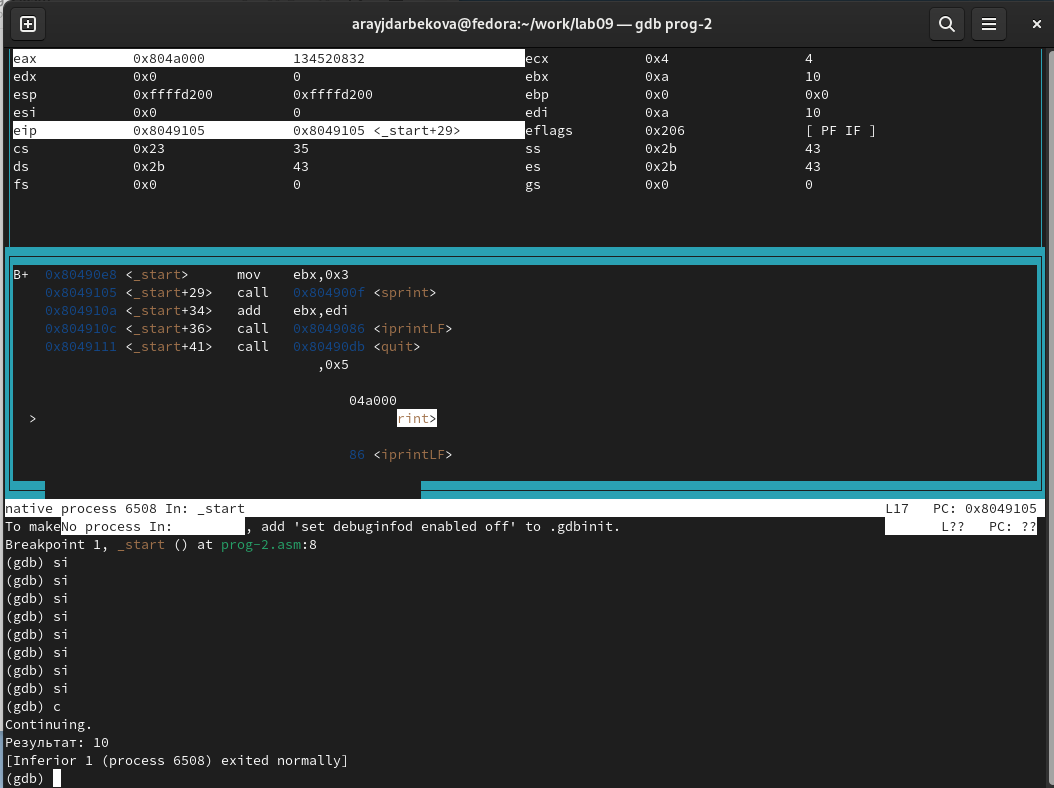


Figure 19: Отладка

Я исправила код программы, учитывая перепутанный порядок аргументов у инструкции add и правильную передачу значения в регистр edi по окончании работы программы. (рис. [[20](#fig:020)]) (рис. [[21](#fig:021)])

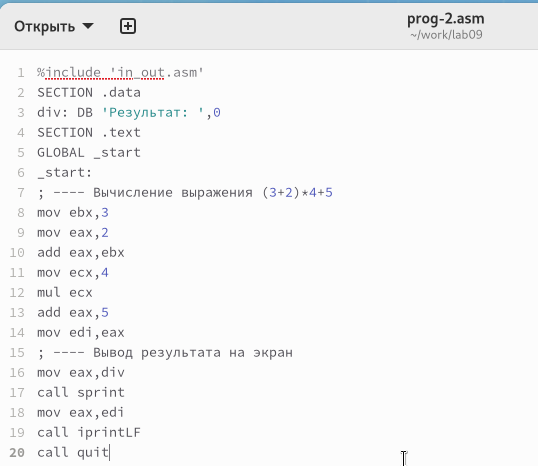


Figure 20: Код исправлен

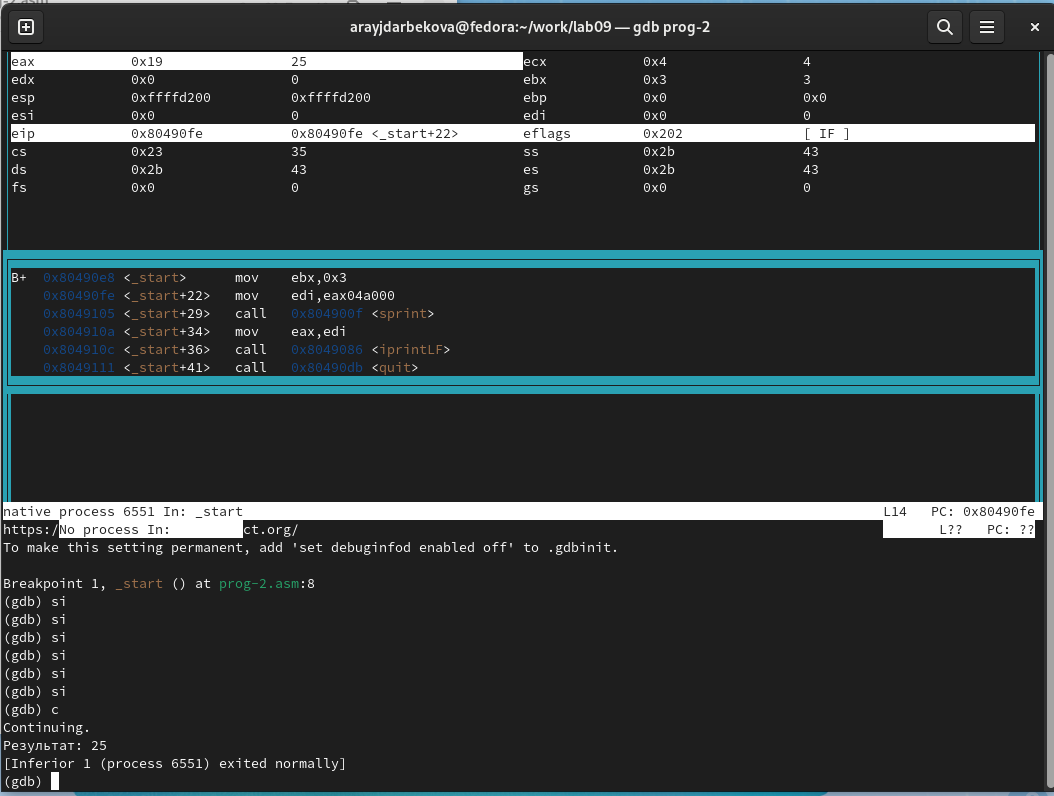


Figure 21: Проверка работы

# 5 Выводы

Освоили работy с подпрограммами и отладчиком.