Отчёт по лабораторной работе 9

Архитектура компьютера

Душаев Азимбек Юсуфович НКАбд-02-23

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 Выполнение лабораторной работы

1. Создал каталог для выполнения лабораторной работы № 9 и перешел в него. Затем создал файл lab9-1.asm.
2. В качестве примера рассмотрим программу, которая вычисляет арифметическое выражение с помощью подпрограммы calcul. В данном примере значение вводится с клавиатуры, а само выражение вычисляется внутри подпрограммы.

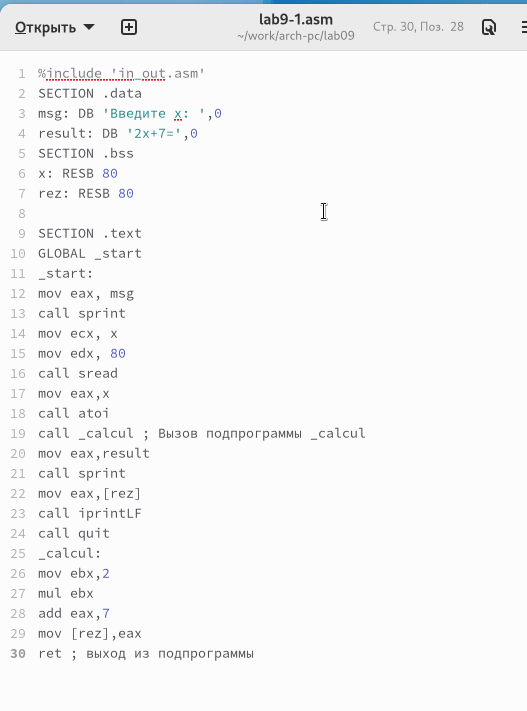


Figure 1: Программа lab9-1.asm

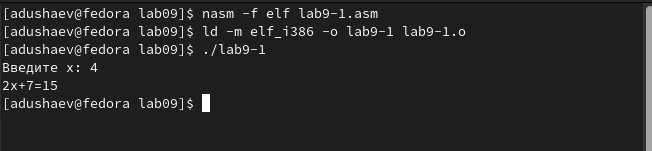


Figure 2: Запуск программы lab9-1.asm

1. Внесены изменения в текст программы, добавлена подпрограмма subcalcul внутри подпрограммы calcul для вычисления выражения , где значение также вводится с клавиатуры, а функции и вычисляются внутри подпрограмм.

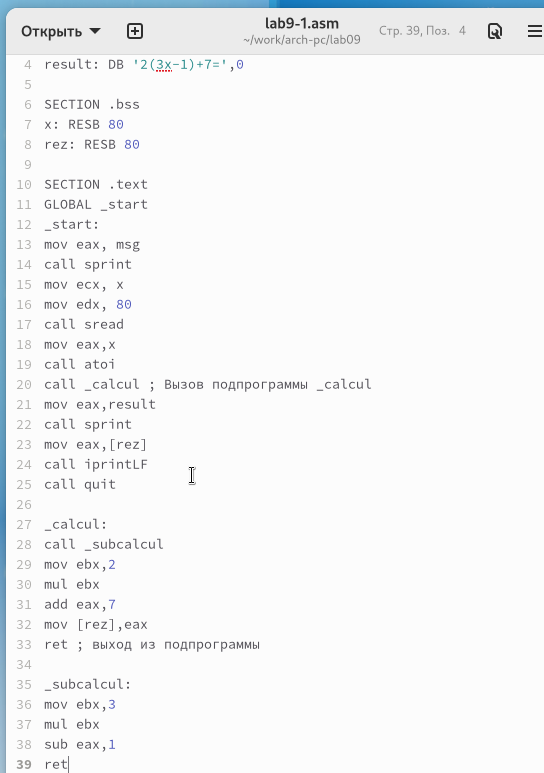


Figure 3: Программа lab9-1.asm

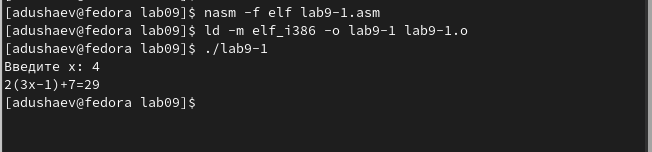


Figure 4: Запуск программы lab9-1.asm

1. Создан файл lab9-2.asm с текстом программы из Листинга 9.2, который представляет программу печати сообщения “Hello world!”.

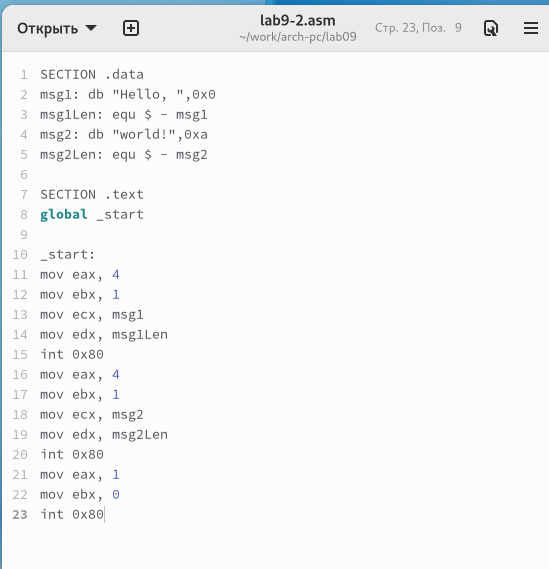


Figure 5: Программа lab9-2.asm

Был получен исполняемый файл. Для возможности работы с отладчиком GDB необходимо добавить отладочную информацию в исполняемый файл, для этого трансляция программы должна быть выполнена с использованием ключа “-g”.

Загрузка исполняемого файла в отладчик GDB была выполнена успешно. Для проверки работоспособности программы была использована команда run (сокращенно r) в оболочке GDB.

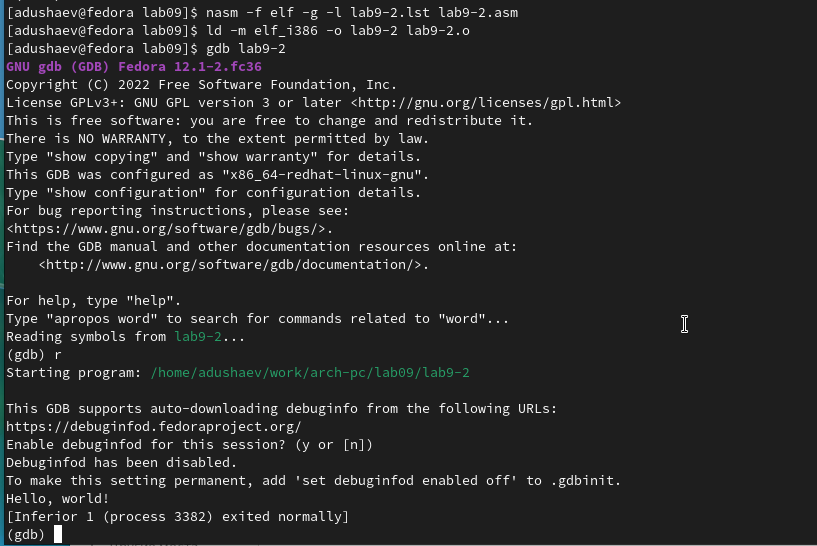


Figure 6: Запуск программы lab9-2.asm в отладчике

Для более детального анализа программы, установите точку остановки на метке “start”, с которой начинается выполнение любой ассемблерной программы, и запустите программу. Затем просмотрите дизассемблированный код программы.

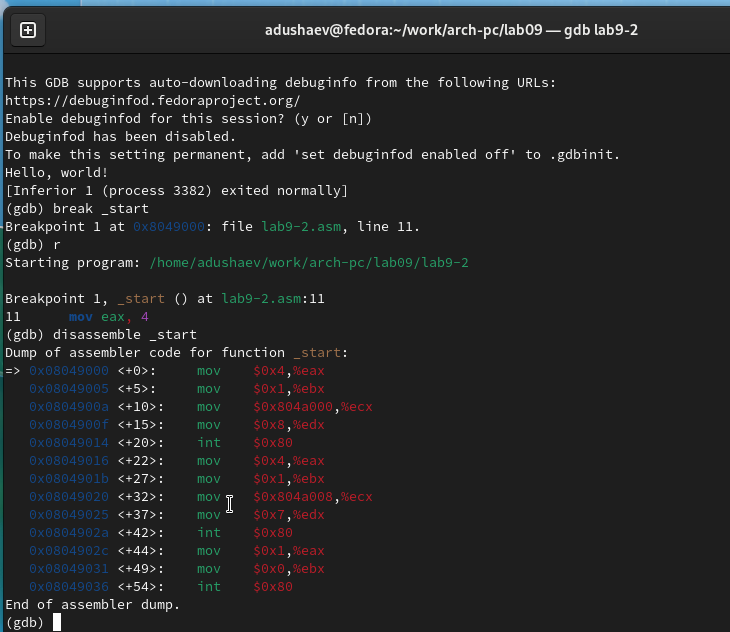


Figure 7: Дизассемблированный код

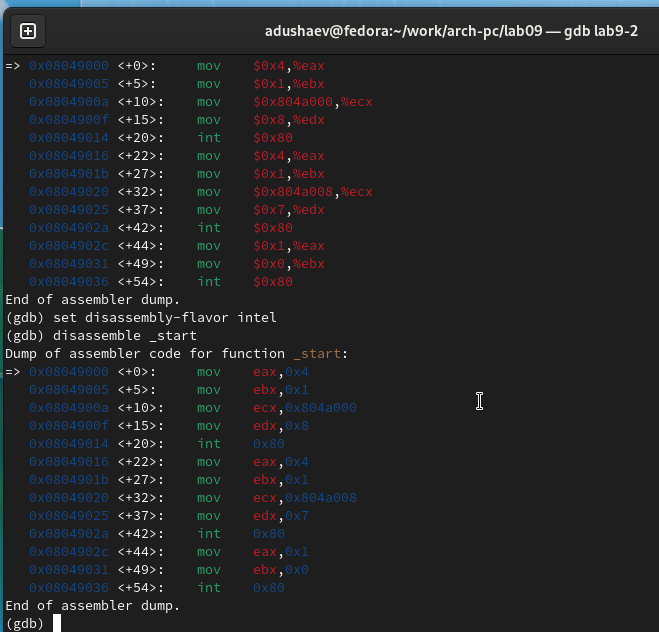


Figure 8: Дизассемблированный код в режиме интел

На предыдущих шагах была установлена точка остановки на метке “\_start”. Чтобы проверить это, воспользуйтесь командой “info breakpoints” (сокращенно “i b”). Добавьте еще одну точку остановки на адресе инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции. Определите адрес предпоследней инструкции “mov ebx, 0x0” и установите точку остановки.

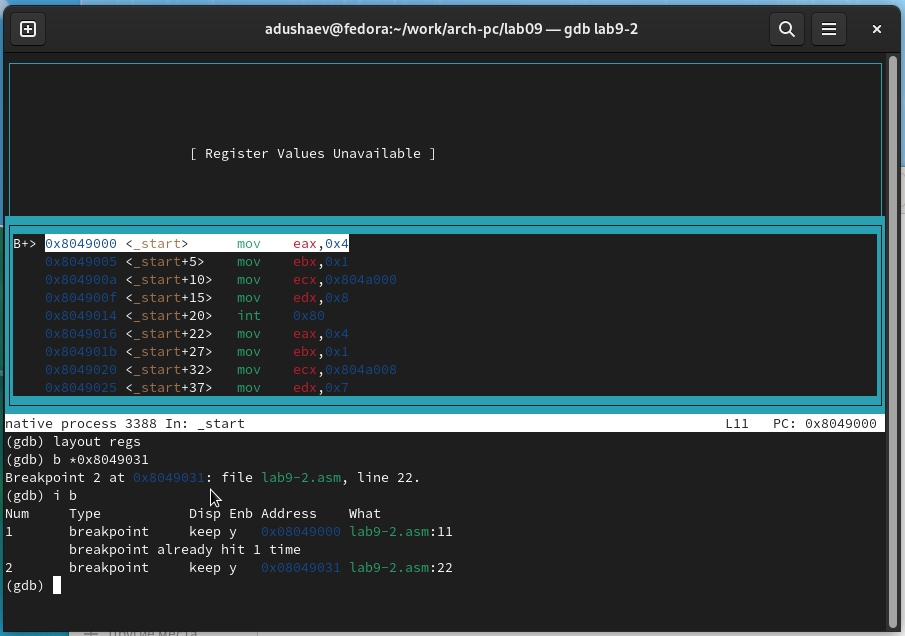


Figure 9: Точка остановки

Отладчик может отображать содержимое ячеек памяти и регистров, а также позволяет вручную изменять значения регистров и переменных. Выполните 5 инструкций с помощью команды “stepi” (или “si”) и проследите за изменением значений регистров.

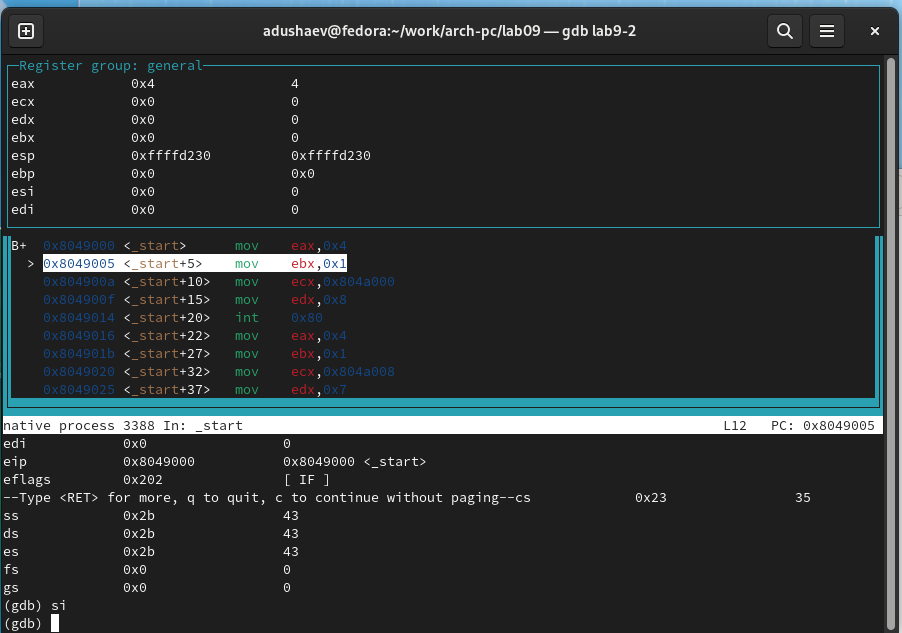


Figure 10: Изменение регистров

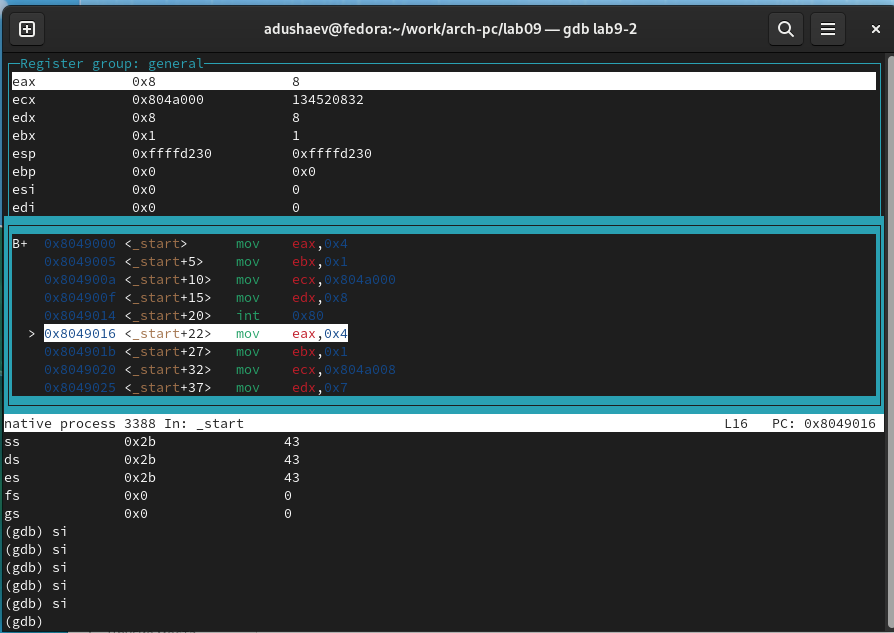


Figure 11: Изменение регистров

Просмотрите значение переменной “msg1” по имени и значение переменной “msg2” по адресу.

Чтобы изменить значение регистра или ячейки памяти, воспользуйтесь командой “set”, указав имя регистра или адрес в качестве аргумента. Измените первый символ переменной “msg1”.

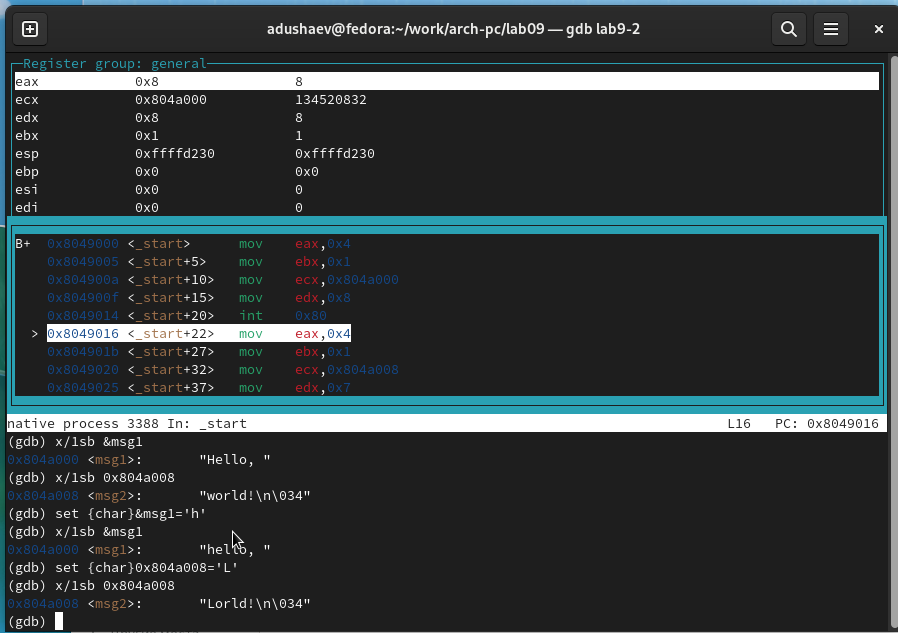


Figure 12: Изменение значения переменной

Выведите значение регистра “edx” в различных форматах (шестнадцатеричном, двоичном и символьном).

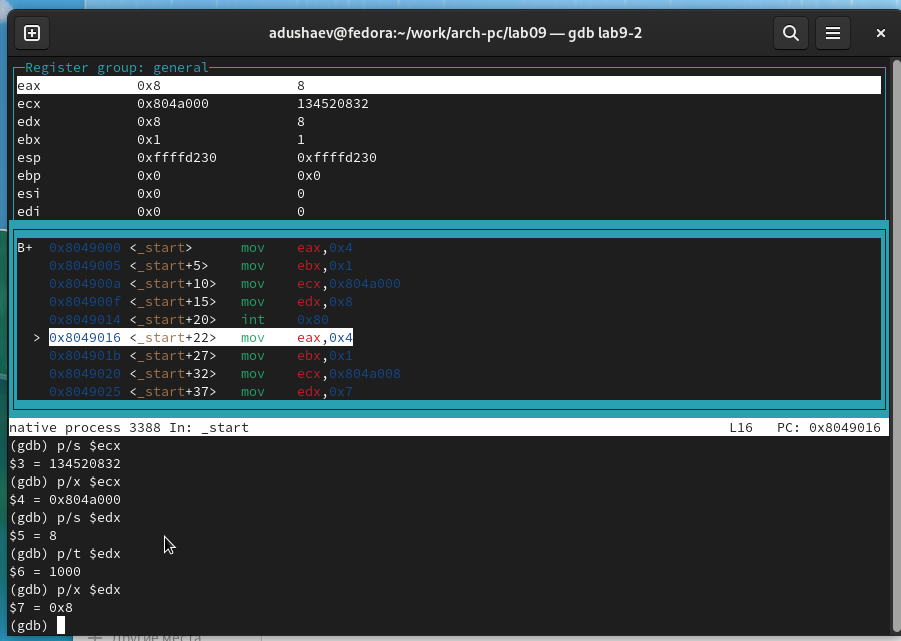


Figure 13: Вывод значения регистра

Используя команду “set”, измените значение регистра “ebx”.

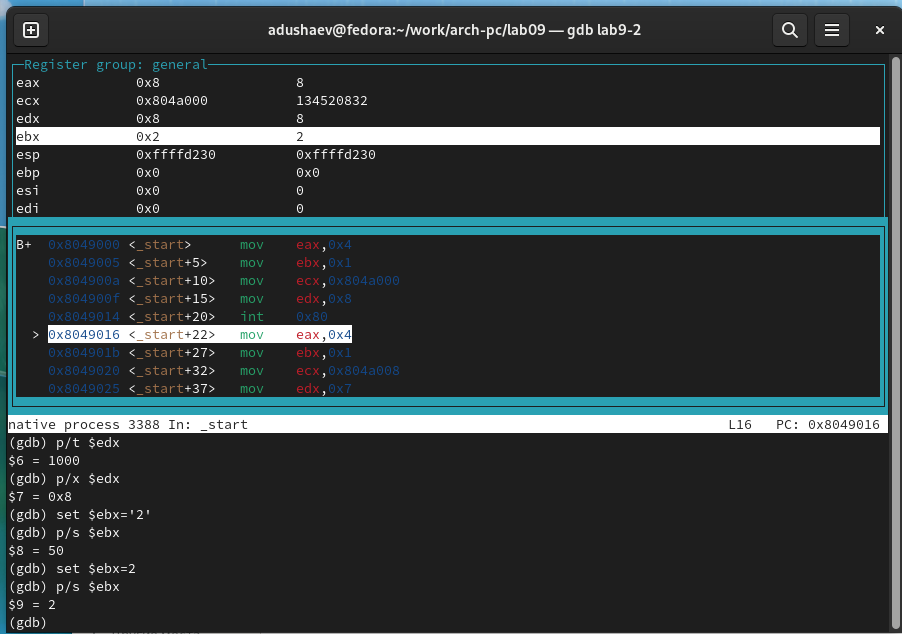


Figure 14: Вывод значения регистра

1. Скопировал файл lab8-2.asm, созданный в процессе выполнения лабораторной работы №8, который содержит программу для вывода аргументов командной строки. Затем создал исполняемый файл. Для загрузки программы с аргументами в отладчик GDB необходимо использовать ключ “–args”. Загрузил исполняемый файл в отладчик, указав аргументы командной строки.

Для начала установил точку останова перед первой инструкцией в программе и запустил ее.

Адрес вершины стека, который хранится в регистре ESP, указывает на число аргументов командной строки (включая имя программы), которое находится по этому адресу. В данном случае число аргументов равно 5, включая имя программы “lab9-3” и сами аргументы: “аргумент1”, “аргумент2” и “аргумент3”.

Далее были просмотрены остальные позиции в стеке. 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 аргумента и так далее.

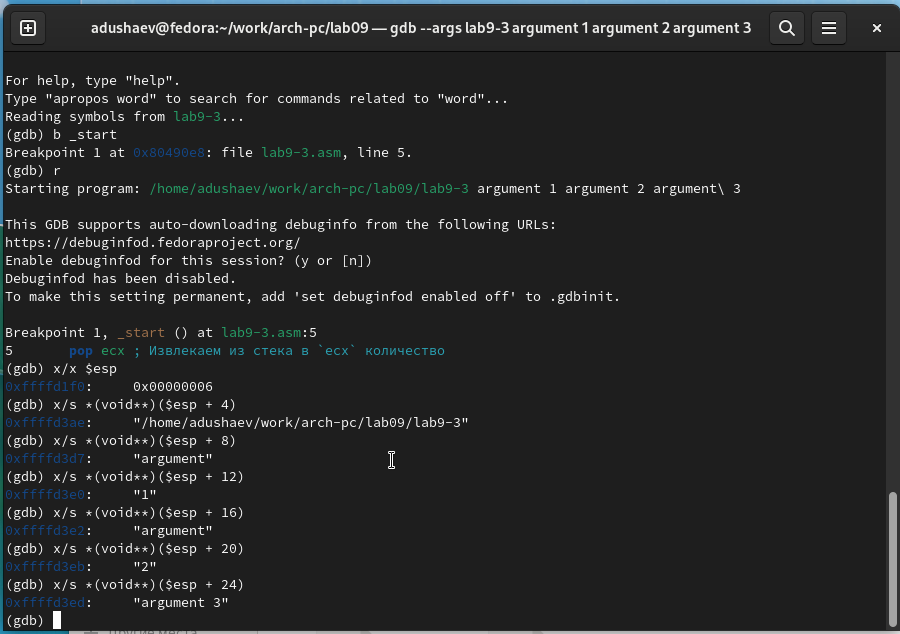


Figure 15: Вывод значения регистра

Шаг изменения адреса равен 4, так как каждая следующая позиция в стеке смещается на размер переменной, который составляет 4 байта.

1. Внес изменения в программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), чтобы реализовать вычисление значения функции f(x) в виде подпрограммы.

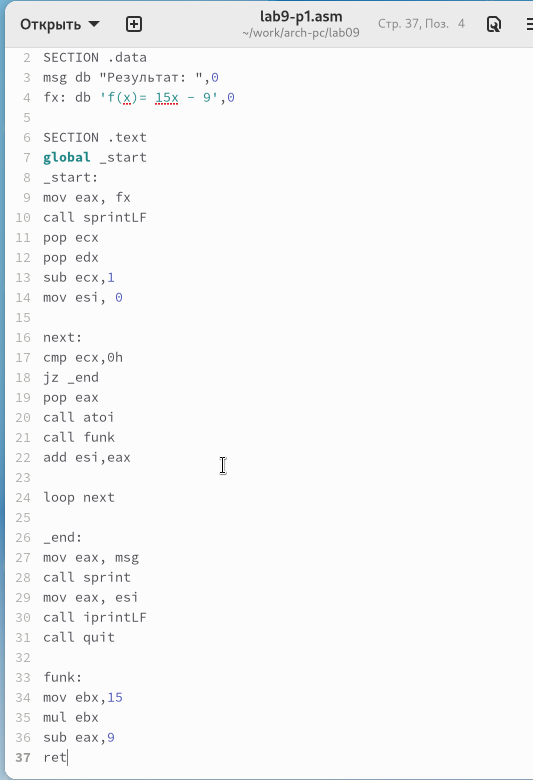


Figure 16: Программа prog1.asm

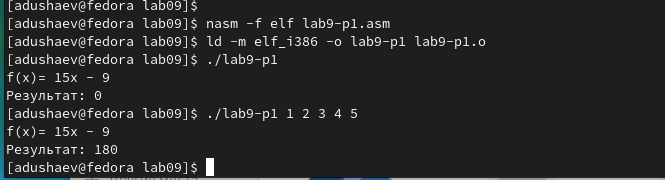


Figure 17: Запуск программы prog1.asm

1. Приведенный ниже листинг содержит программу для вычисления выражения . После запуска данная программа дает неверный результат. Проверил это с помощью отладчика GDB и анализа изменений значений регистров. Определил ошибку и внес соответствующие исправления

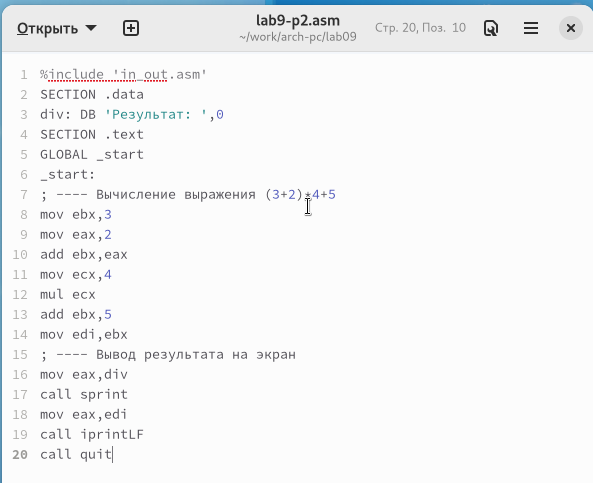


Figure 18: Код с ошибкой

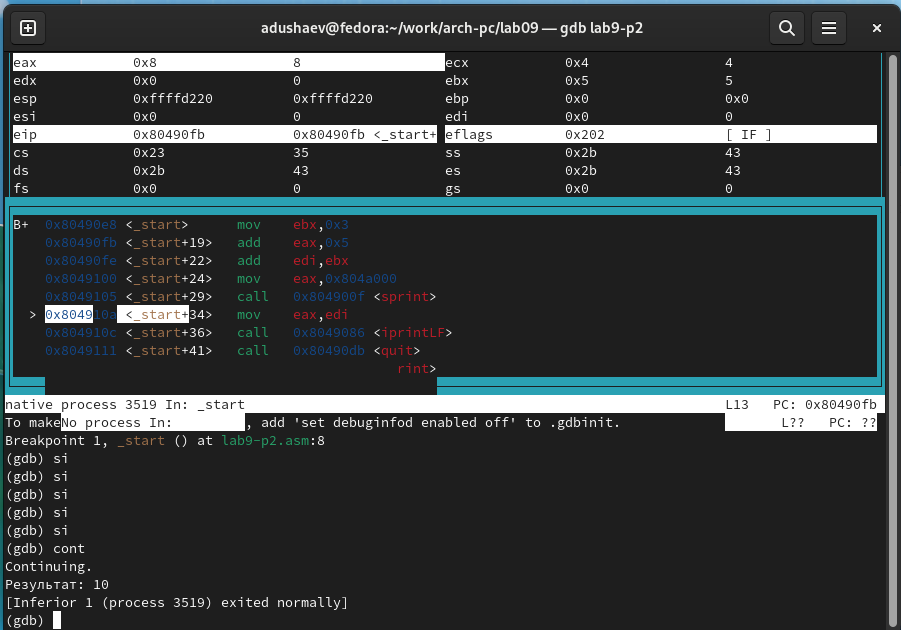


Figure 19: Отладка

Отмечу, что в исходной программе был перепутан порядок аргументов у инструкции add, а также в конце работы значение ebx передается в edi вместо eax.

Исправленный код программы

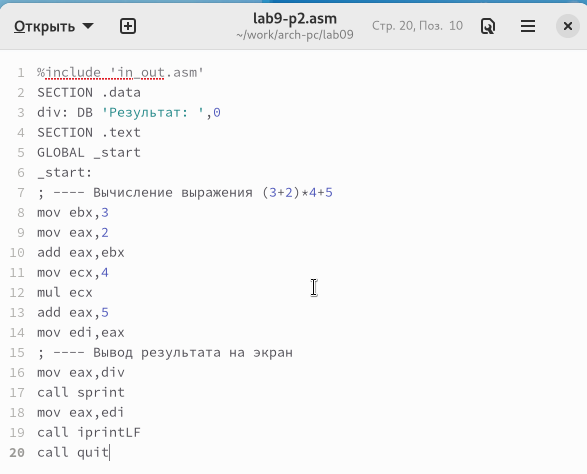


Figure 20: Код исправлен

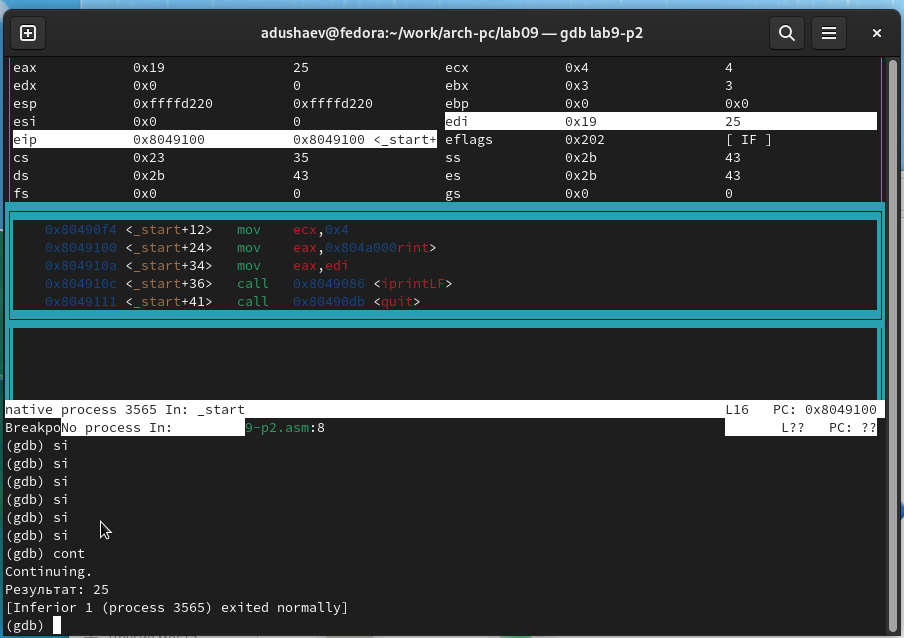


Figure 21: Проверка работы

# 3 Выводы

Освоили работy с подпрограммами и отладчиком.