Отчёт по лабораторной работе 9

Архитектура компьютера

Эргешов Байрам 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

Я создал каталог для выполнения лабораторной работы № 9 и перешел в него. Затем я создал файл lab9-1.asm.

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.

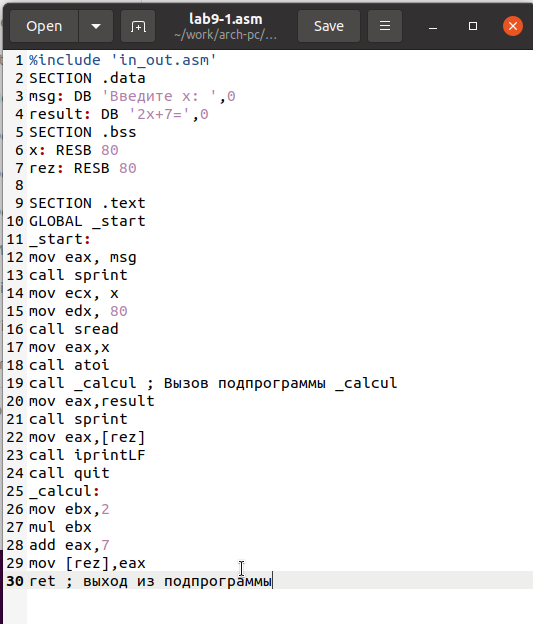


Figure 1: Код программы lab9-1.asm

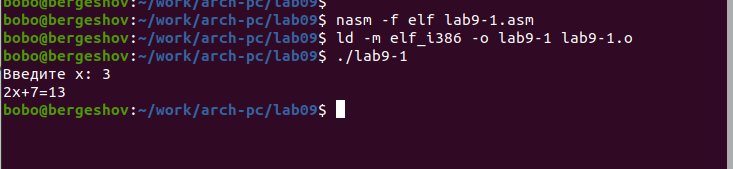


Figure 2: Компиляция и запуск программы lab9-1.asm

Далее, я внес изменения в текст программы, добавив подпрограмму subcalcul внутри подпрограммы calcul для вычисления выражения , где значение также вводится с клавиатуры, а функции и вычисляются внутри подпрограмм.

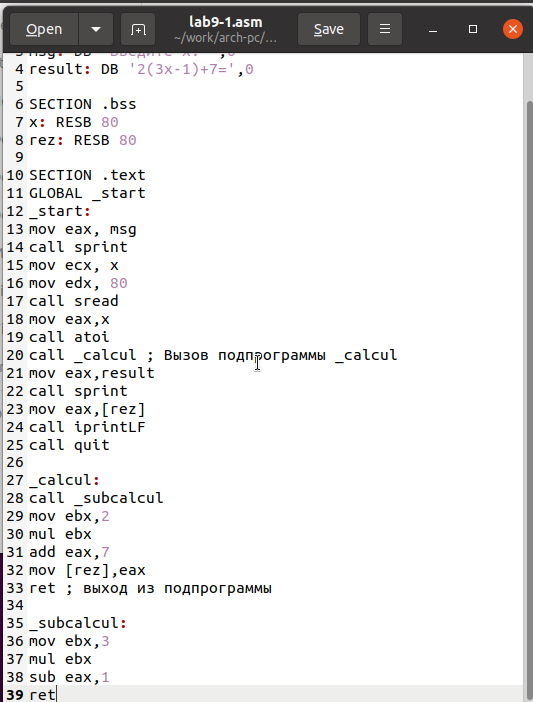


Figure 3: Код программы lab9-1.asm

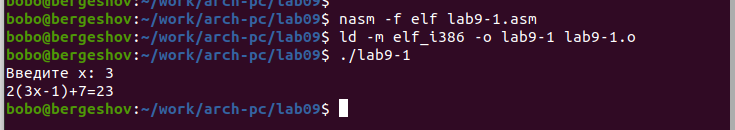


Figure 4: Компиляция и запуск программы lab9-1.asm

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

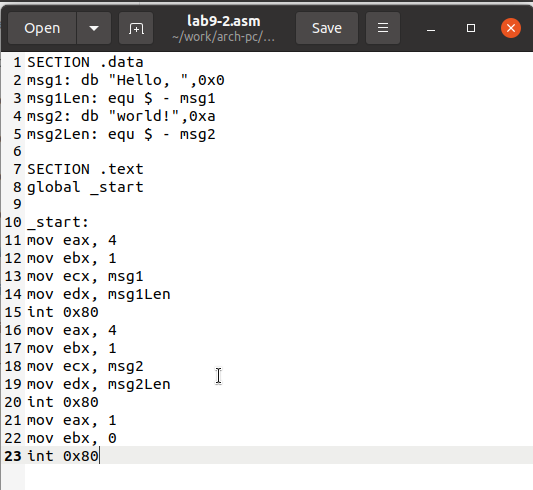


Figure 5: Код программы lab9-2.asm

Получив исполняемый файл, я добавил отладочную информацию с помощью ключа ‘-g’ для возможности работы с отладчиком GDB.

Затем я загрузил исполняемый файл в отладчик GDB и проверил работу программы, запустив ее с помощью команды ‘run’ (сокращенно ‘r’).

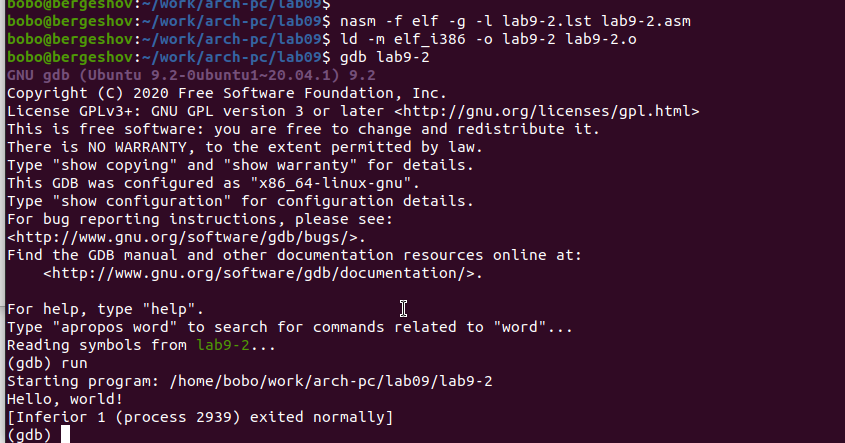


Figure 6: Компиляция и запуск программы lab9-2.asm в отладчике

Для более детального анализа программы, я установил точку остановки на метке ‘start’, с которой начинается выполнение любой ассемблерной программы, и запустил ее. Затем я просмотрел дизассемблированный код программы.

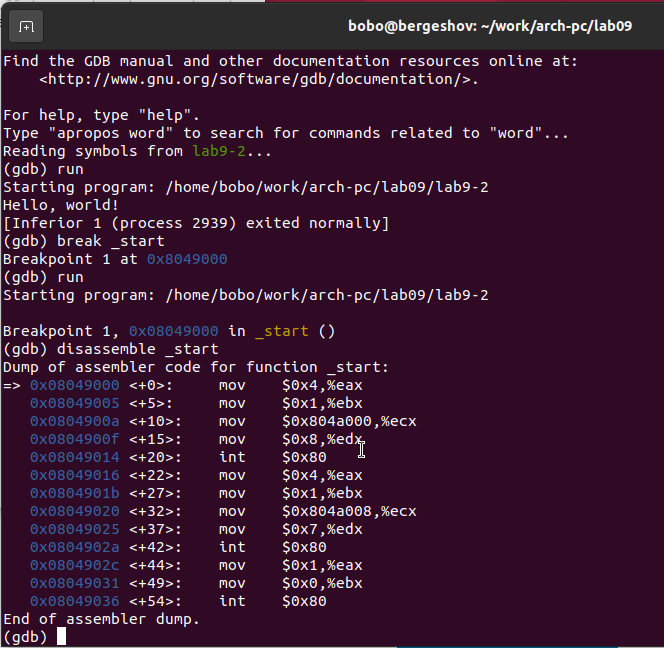


Figure 7: Дизассемблированный код



Figure 8: Дизассемблированный код в режиме интел

Для проверки точки остановки по имени метки ’\_start’, я использовал команду ‘info breakpoints’ (сокращенно ‘i b’). Затем я установил еще одну точку остановки по адресу инструкции, определив адрес предпоследней инструкции ‘mov ebx, 0x0’.

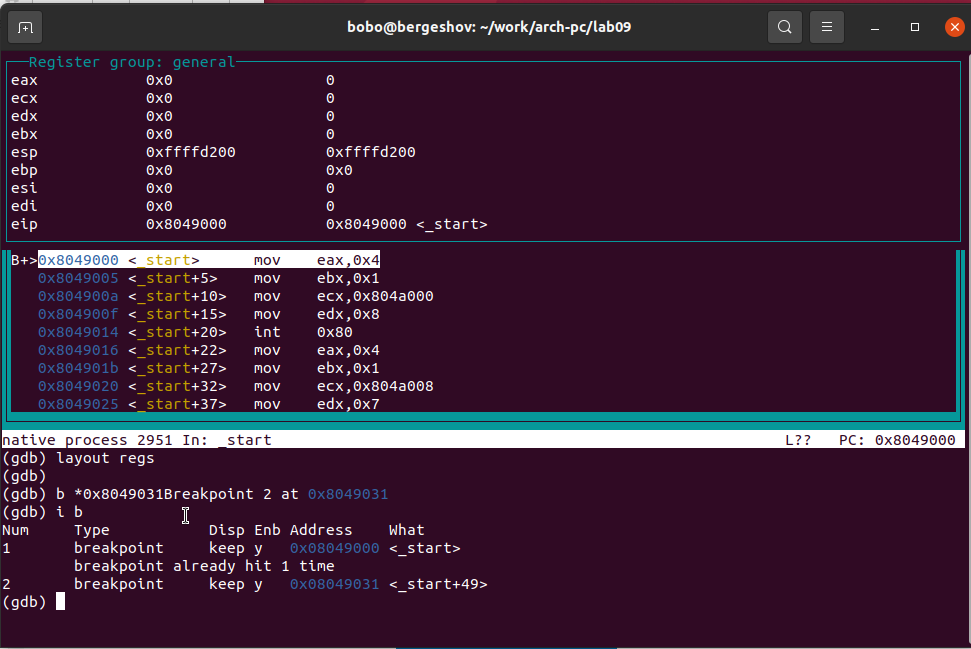


Figure 9: Точка остановки

В отладчике GDB я имел возможность просматривать содержимое ячеек памяти и регистров, а также изменять значения регистров и переменных. Я выполнил 5 инструкций с помощью команды ‘stepi’ (сокращенно ‘si’) и отследил изменение значений регистров.

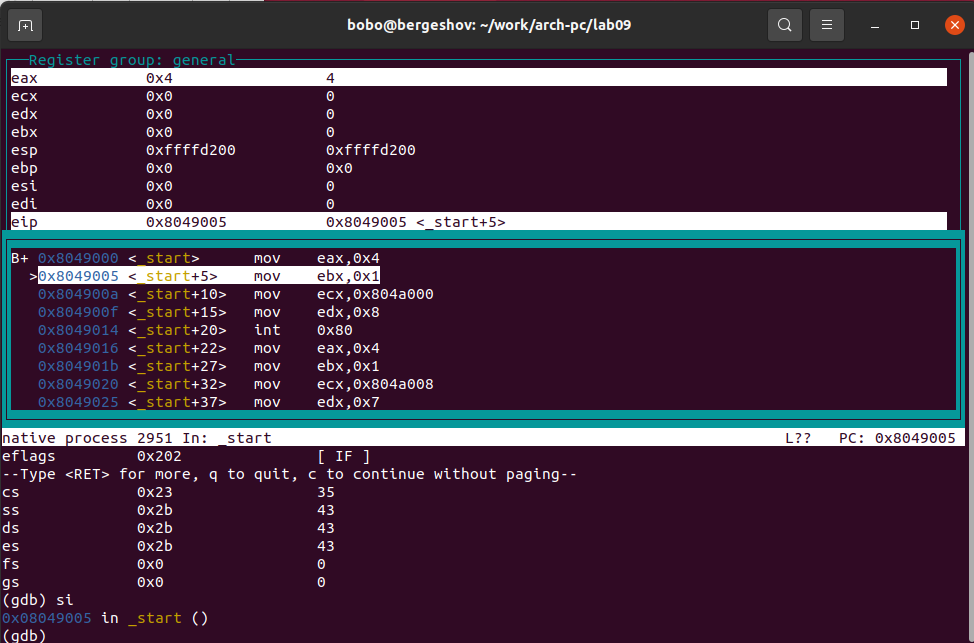


Figure 10: Изменение регистров

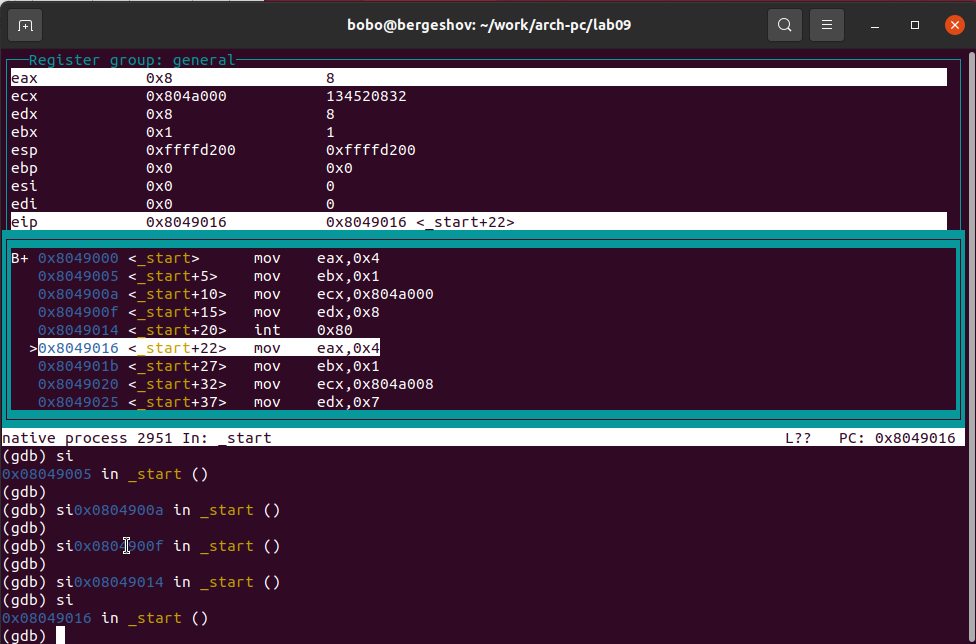


Figure 11: Изменение регистров

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

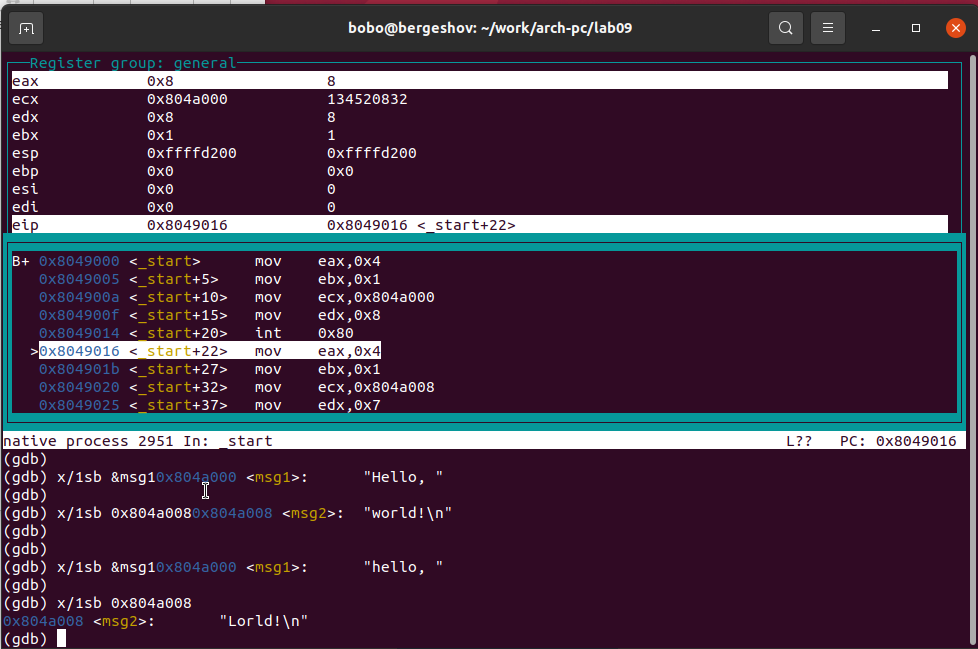


Figure 12: Изменение значения переменной

Также, для изменения значения регистра или ячейки памяти, я использовал команду ‘set’, указав имя регистра или адрес в качестве аргумента. Я успешно изменил первый символ переменной msg1.

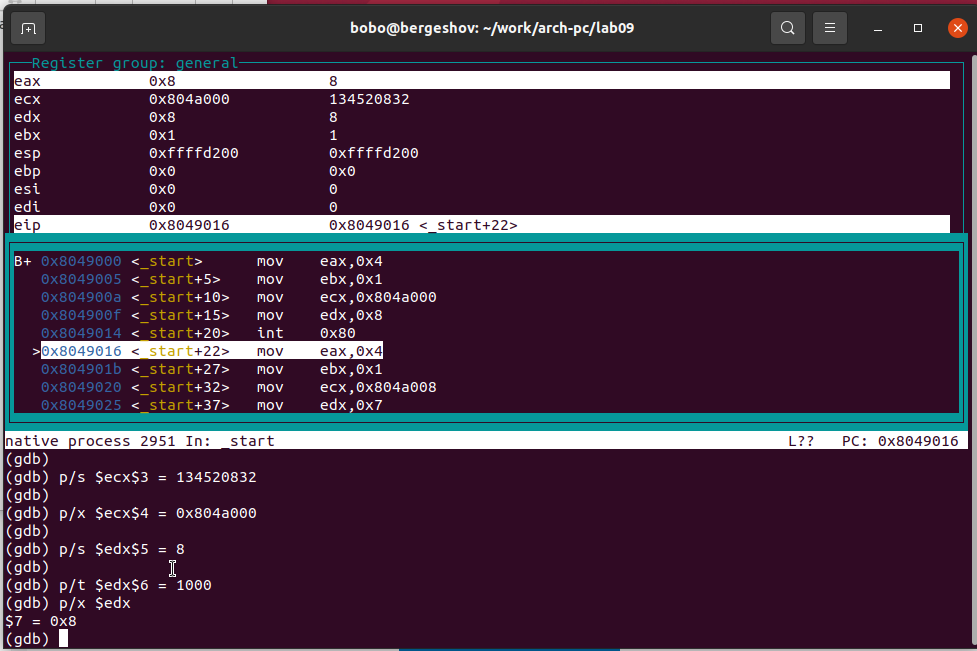


Figure 13: Вывод значения регистра

С помощью команды set изменил значение регистра ebx на нужное значение.

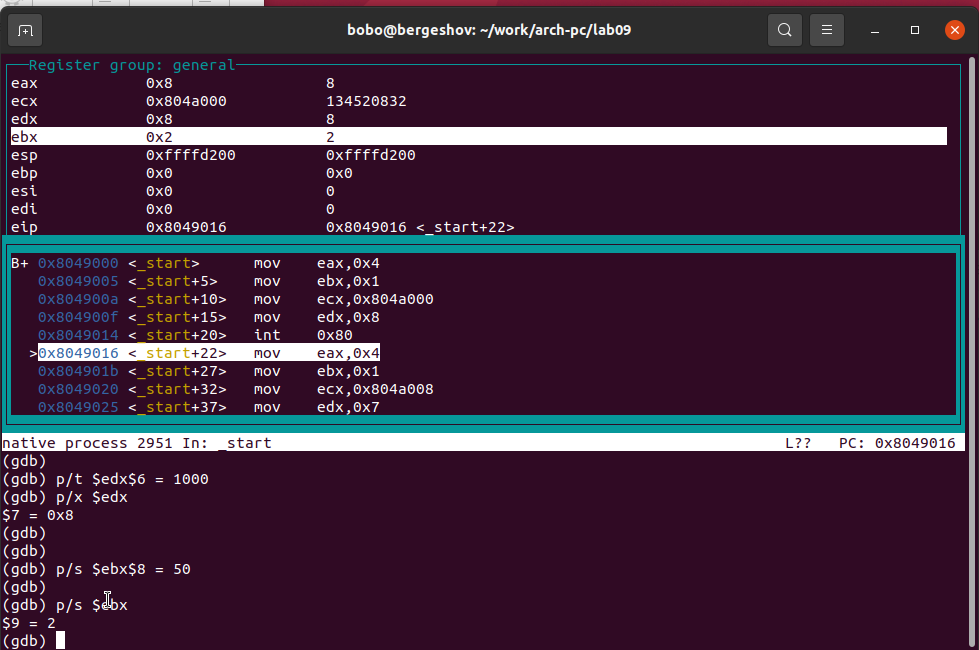


Figure 14: Вывод значения регистра

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

Для загрузки программы с аргументами в отладчик GDB, я использовал ключ –args и загрузил исполняемый файл в отладчик с указанными аргументами.

Я установил точку останова перед первой инструкцией программы и запустил ее.

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

Я также просмотрел 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]).

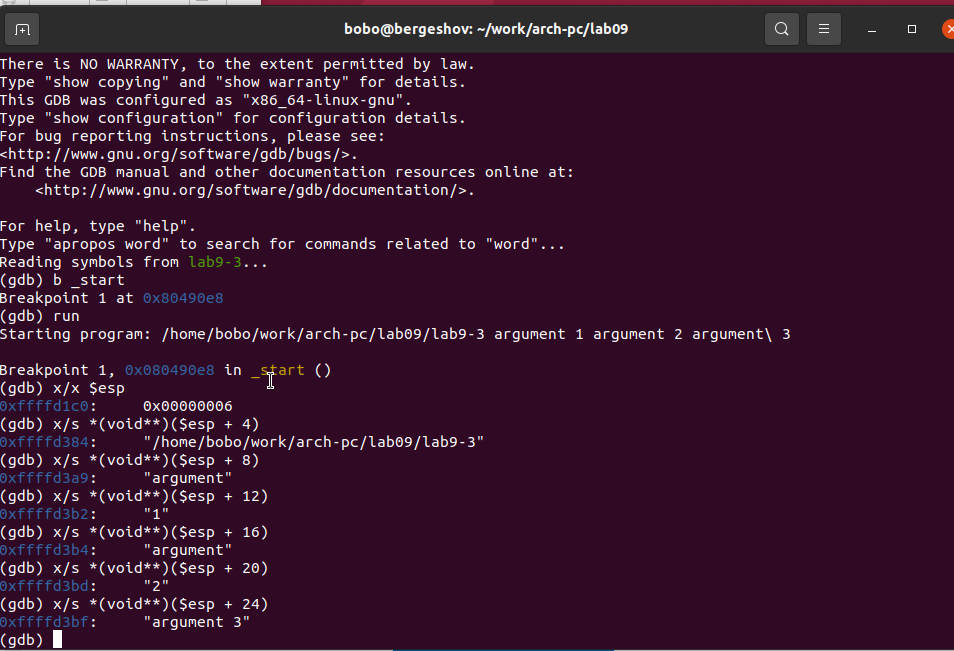


Figure 15: Вывод значения регистра

## 2.1 Задание для самостоятельной работы

Преобразовал программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции f(x) как подпрограмму.

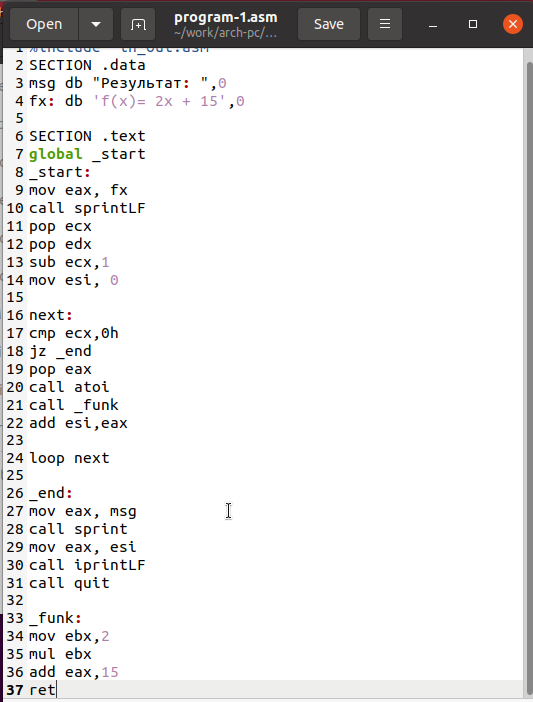


Figure 16: Код программы program-1.asm



Figure 17: Компиляция и запуск программы program-1.asm

В листинге приведена программа вычисления выражения . При запуске данная программа дает неверный результат. Проверил это, анализируя изменения значений регистров с помощью отладчика GDB.

Определил ошибку - перепутан порядок аргументов у инструкции add. Также обнаружил, что по окончании работы в edi отправляется ebx вместо eax.

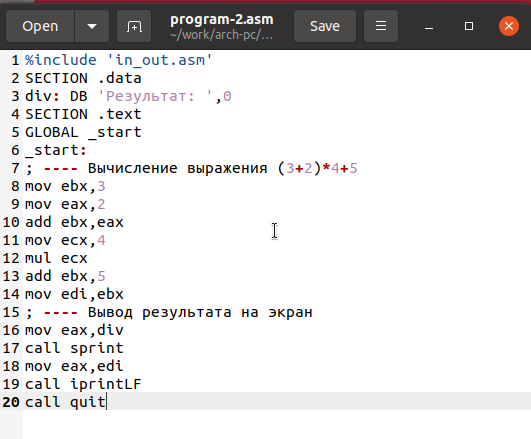


Figure 18: Код программы program-2.asm с ошибкой

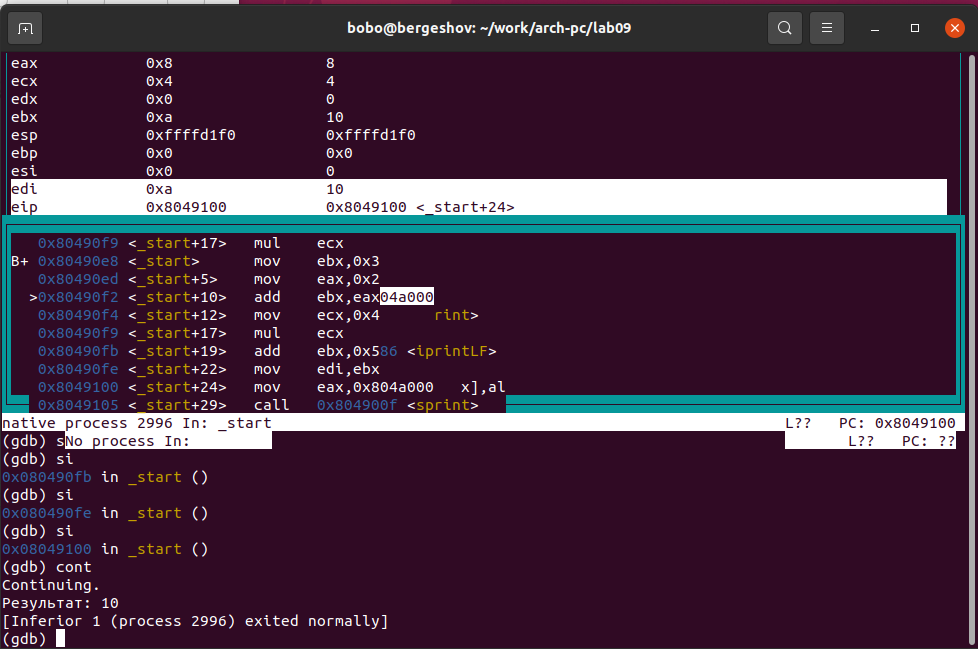


Figure 19: Отладка

Отмечу, что перепутан порядок аргументов у инструкции add и что по окончании работы в edi отправляется ebx вместо eax

Исправленный код программы

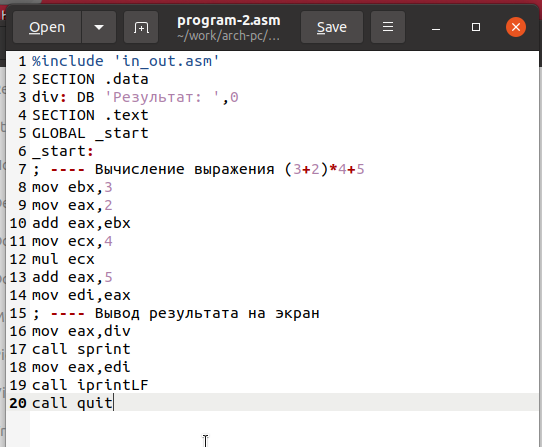


Figure 20: Код программы program-2.asm исправлен

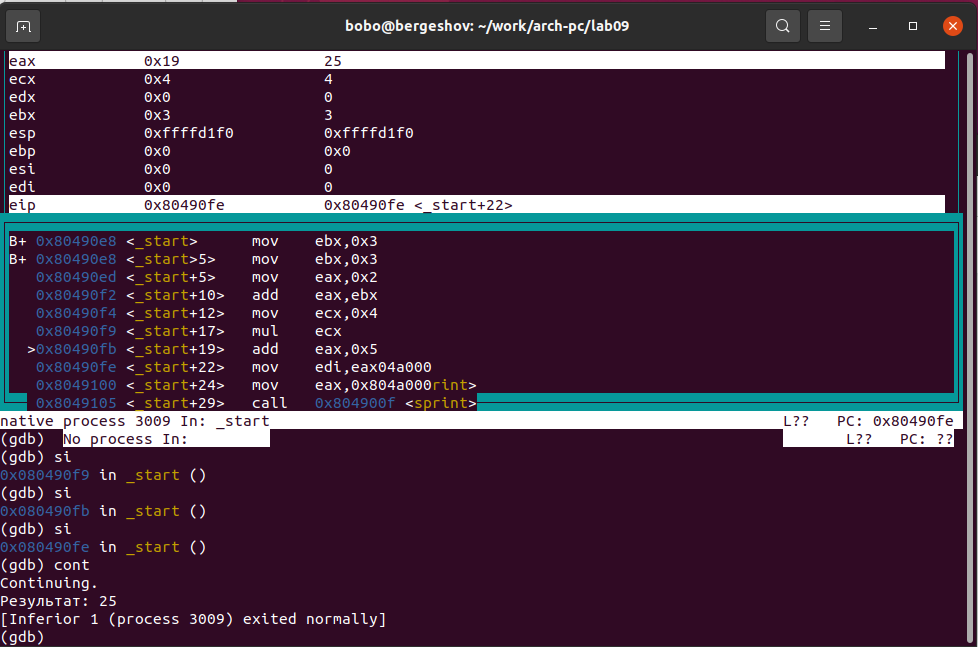


Figure 21: Проверка работы

# 3 Выводы

Освоили работy с подпрограммами и отладчиком.