Лабораторная работа №9

Понятие подпрограммы.Отладчик GDB.

Прозорова Елизавета Евгеньевна

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

1.Сначала я создала каталог для программам лабораторной работы № 9, затем перешла в него и создала файл lab9-1.asm

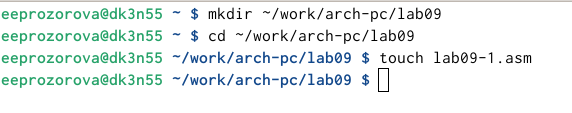


Рис. 1: Создание каталога и файла lab9-1

1. Я ввела в файл lab09-1.asm текст программы из листинга 9.1.

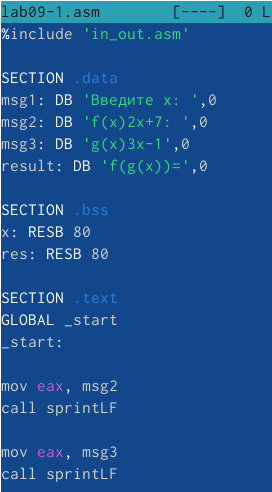


Рис. 2: Текст программы lab09-1

Я создала исполняемый файл и запустила его. Для x я выбрала числа 2 и 9.

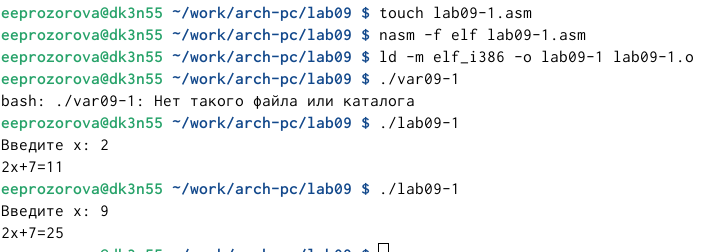


Рис. 3: Создание и запуск lab09-1

Я изменила текст программы, добавив подпрограмму \_subcalcul в подпрограмму \_calcul,для вычисления выражения f(g(x)).

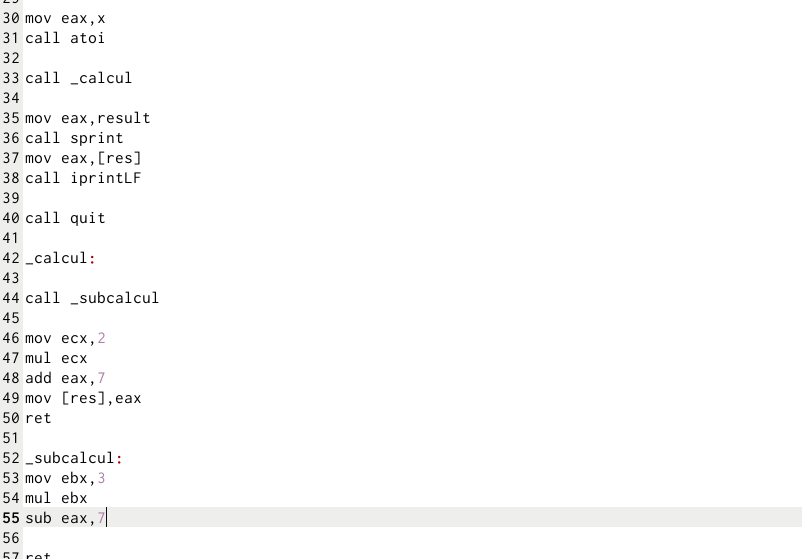


Рис. 4: Изменения текста

Затем я создала и проверила измененный файл.

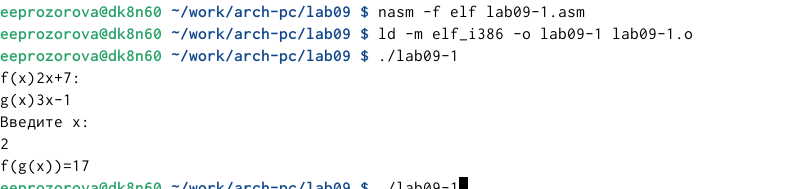


Рис. 5: Создание и запуск lab09-1

1. Я создала файл lab09-2.asm в каталоге ~/work/arch-pc/lab09 и ввела в него текст программы из листинга 9.2.

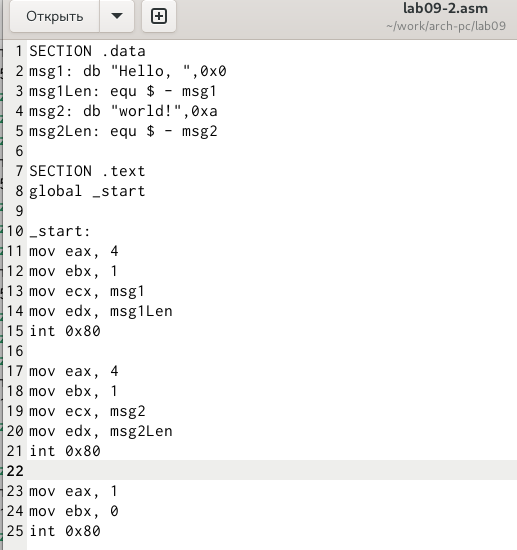


Рис. 6: Текст программы lab09-2.asm

Затем я создала измененный файл ключом ‘-g’.

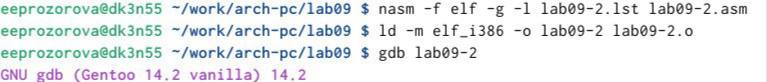


Рис. 7: Получение исполняемого файла

Я загрузила исполняемый файл в отладчик gdb:

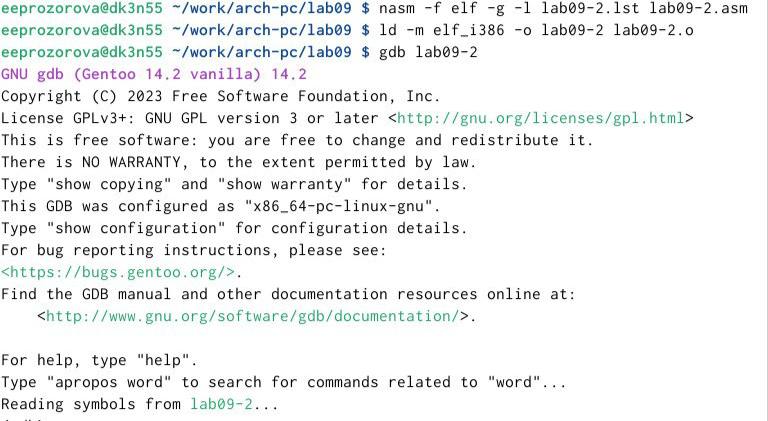


Рис. 8: Отладчик gbd

Затем я проверила работу программы, запустив ее в оболочке GDB с помощью команды run

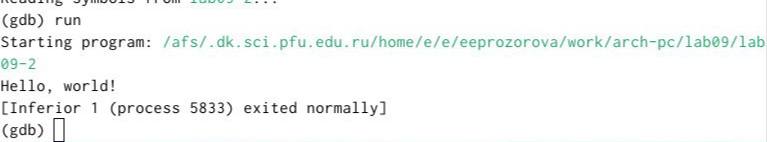


Рис. 9: Проверка работы программы

Для более подробного анализа программы установила брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запустила её.

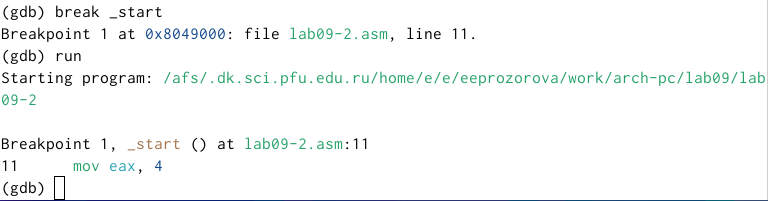


Рис. 10: Утановка брейкпоинта

Затем я посмотрела дисассимилированный код программы



Рис. 11: Команда disassemble

Затем я переключилась на отображение команд с Intel’овским синтаксисом

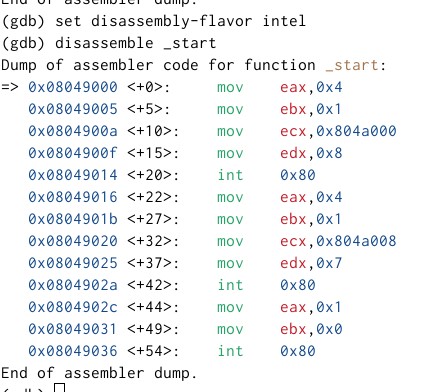


Рис. 12: Intel’овский синтаксис

Различия отображения синтаксиса машинных команд в режимах ATT и Intel:

1. Порядок операндов ATT: источник - назначение (напр. movl %ebx, %eax). Intel: назначение - источник (напр. MOV EAX, EBX).
2. Регистры ATT: С префиксом % (напр. %eax). Intel: Без префикса (напр. EAX).
3. Размеры операндов ATT: Указывается суффикс (b, w, l, q для 1, 2, 4, 8 байт). Intel: Используются указатели (BYTE PTR, DWORD PTR и т.д.).
4. Память ATT: Круглые скобки для указания адреса (напр. 4(%ebx)). Intel: Квадратные скобки (напр. [EBX+4]).

Я включила режим псевдографики для более удобного анализа программы

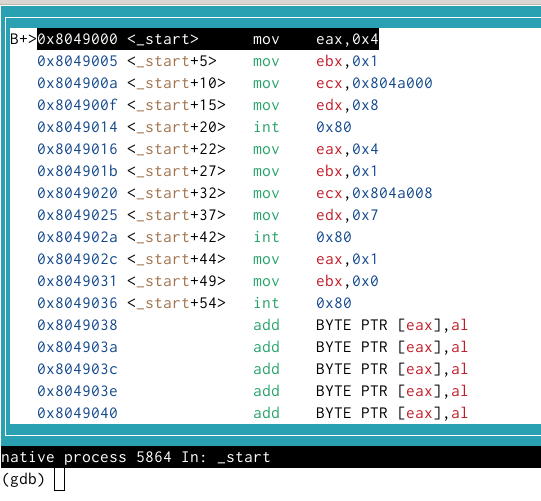


Рис. 13: Команда layout asm

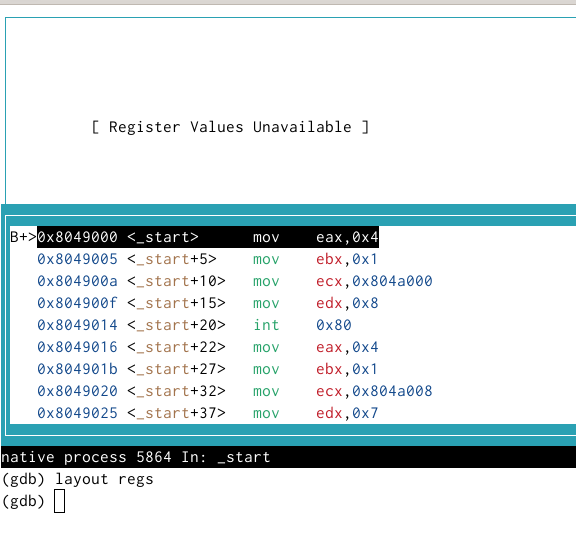


Рис. 14: Команда layout regs

1. Так как на предыдущих шагах была установлена точка останова по имени метки (\_start), я проверю это с помощью команды info breakpoints.

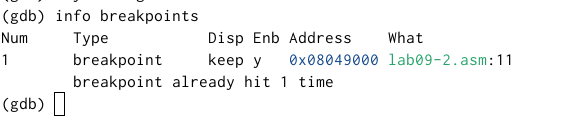


Рис. 15: Команда info breakpoints.

Теперь установим точку остановк на адресе предпоследней инструкции (mov ebx,0x0).

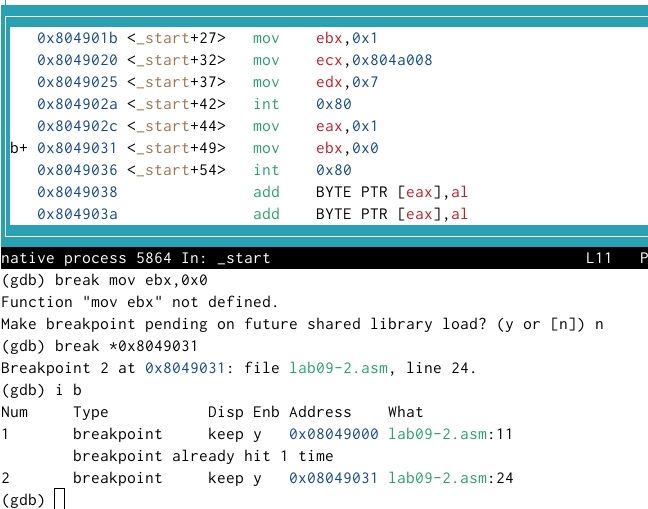


Рис. 16: Установка точки

Посмотрим информацию о всех установленных точках останова.

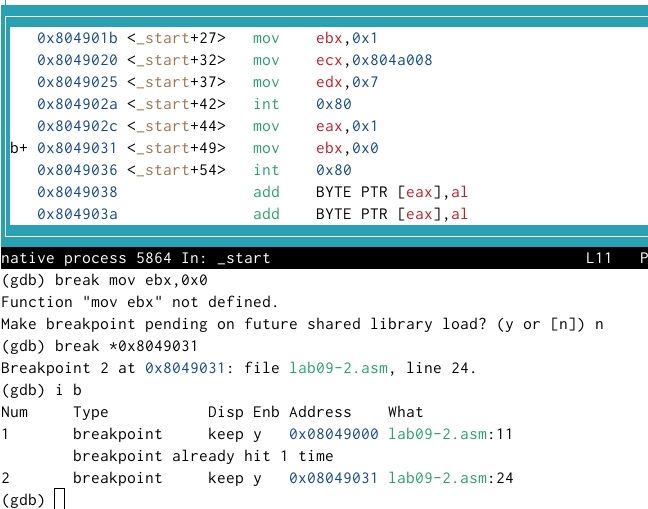


Рис. 17: Команда i b

1. Я посмотрела содержимое регистров также можно с помощью команды info registers

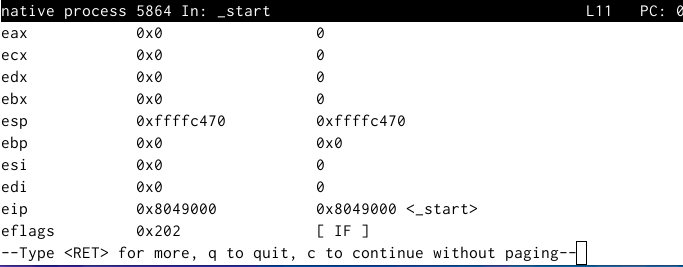


Рис. 18: Команда info registers

Затем я посмотрела значение переменной msg1 по имени.



Рис. 19: Значение переменной msg1 по имени

я посмотрела значение переменной msg2 по адресу.

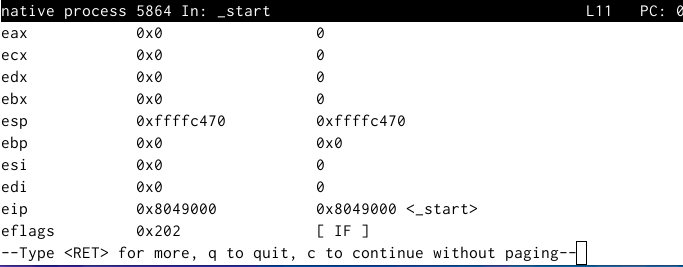


Рис. 20: Значение переменной msg2 по адресу

Изменила первый символ переменной msg1. Теперь первая буква Hello стала маленькой.

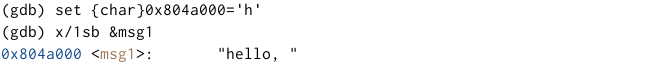


Рис. 21: Команда изменения значение для регистра msg1

Теперь заменю первый и второй символы переменной msg2 на P и a соответственно.

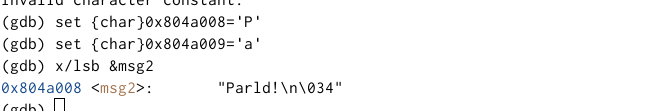


Рис. 22: Команда изменения значение для регистра msg2

С помощью команды print вывела в различный форматах значение регистра edx.



Рис. 23: регистр edx в различный форматах

С помощью команды set изменила значение регистра ebx



Рис. 24: Команда set

Разница выводов команд p/s $ebx: В первом случае регистор хранит ASCII-код символа и gdb пытается интерепретировать его как строку. Во втором случае $ebx хранит число и gdb пытается интерпретировать его как адрес, что выводит ошибку

Затем я завершила выволнение програмы и вышла из GDB.

Рис. 25: Завершение выволнения програмы и выход из GDB.

Рис. 25: Завершение выволнения програмы и выход из GDB.

1. Я скопировала файл lab8-2.asm в файл с именем lab09-3.asm

Рис. 26: Копирование файла lab8-2.asm

Рис. 26: Копирование файла lab8-2.asm

Я создала исполняемый файл.

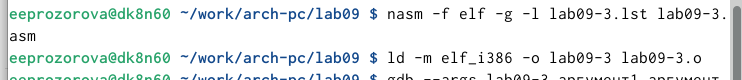


Рис. 27: Создания исполняемого файла

Загрузила исполняемый файл в отладчик, указав аргументы ргумент1 аргумент 2 ’аргумент 3

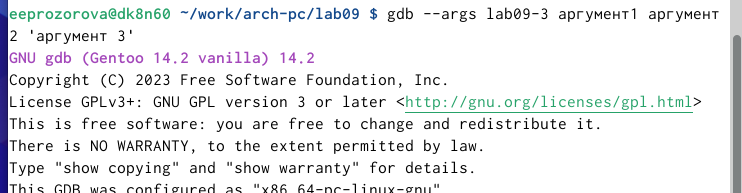


Рис. 28: загрузка в gdb программы с аргументами

Установила точку останова перед первой инструкцией в программе и запустила ее.

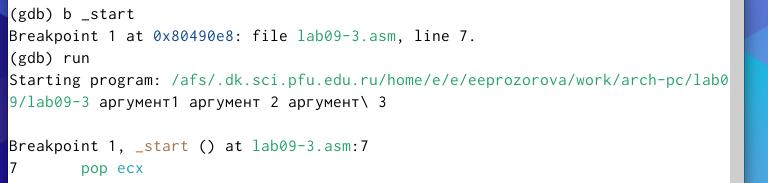


Рис. 29: Установка точки останова

Я вывела адрес вершины стека

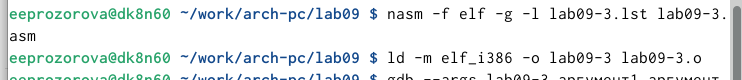


Рис. 30: Вывод адрес вершины стека

Затем я посмотрела остальные позиции стека

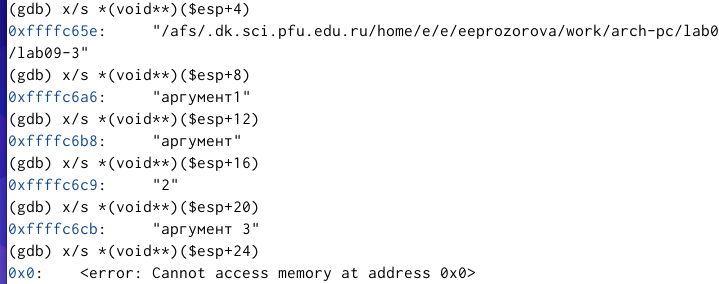


Рис. 31: Позиции стека

Шаг изменения адреса равен 4, потому что в 32-битной архитектуре размер слова — 4 байта, стек выравнивается по 4 байта, и каждый параметр или локальная переменная занимает 4 байта.

# 3 Выполнение самостоятельной работы

1. Я скопировала программу из лабораторной работы №8, и назвала lab09-4.asm

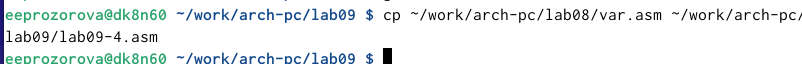


Рис. 32: Копирование программы из лабораторной работы №8

Затем я изменила эту программу, реализовав вычисление значения функции 𝑓(𝑥) как подпрограмму.

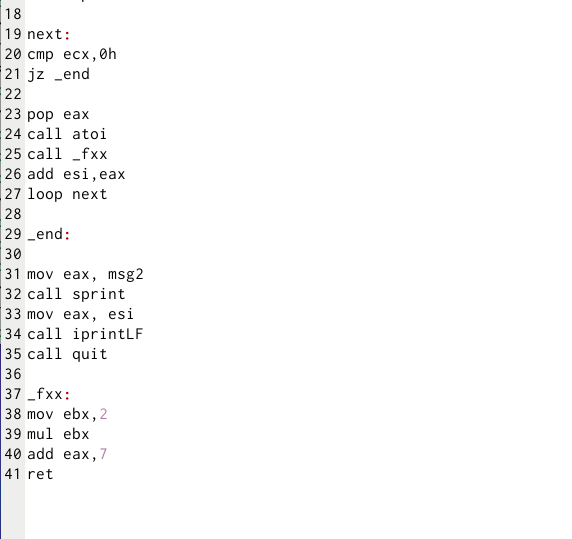


Рис. 33: Измененный текст программы

Затем я создала и проверила измененный файл.

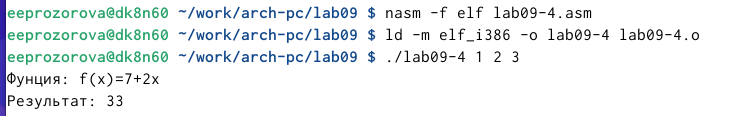


Рис. 34: Проверка программы

1. Я создала файл lab09-5.asm в который написала программу из листинга 9.3. с программа вычисления выражения (3+2)\*4+5

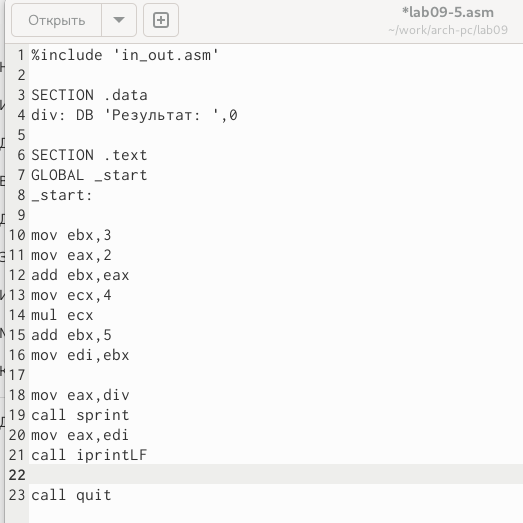


Рис. 35: Текст программы из листинга 9.3.

Я создала и проверила файл. Ответ неверный.

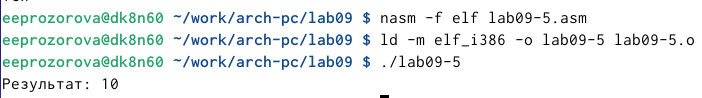


Рис. 36: Проверка программы

Затем я открыла этот файл в GDB.

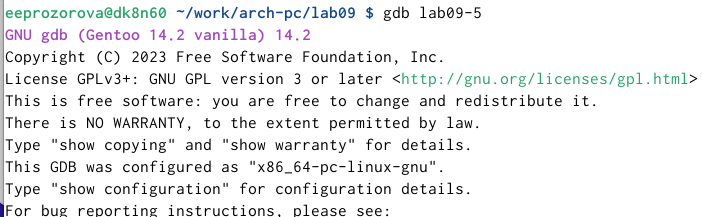


Рис. 37: Файл lab09-5 в GDB

Для более подробного анализа программы я установила брейкпоинт на метку \_start.

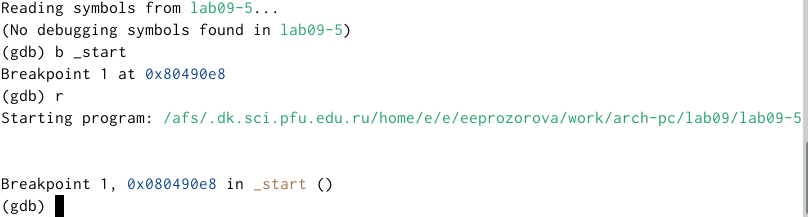


Рис. 38: Установка брейкпоинта

Затем я посмортела дисассимилированный код программы, и переключилась на отображение команд с Intel’овским синтаксисом



Рис. 39: Дисассимилированный код программы

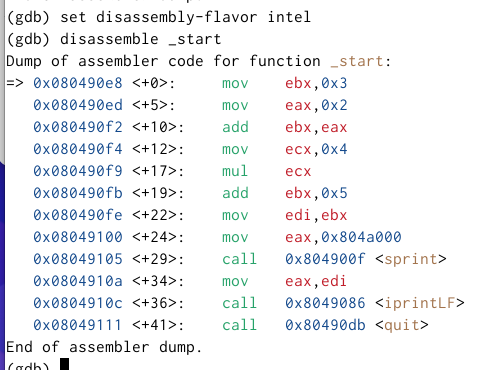


Рис. 40: Отображение команд с Intel’овским синтаксисом

Включила режим псевдографики для более удобного анализа программы



Рис. 41: Режим псевдографики

Затем я подробно изучила все строки программы, и нашла ошибку, в определенном месте значения не соответсветсвовали своим регистрам. Я исправила ошибку, поменяв их местами.

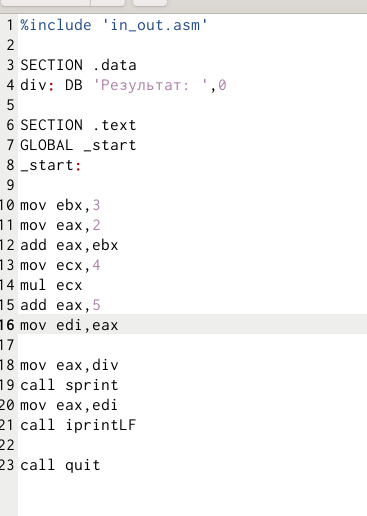


Рис. 42: Измененнй текст программы

Я создала и проверила измененный файл. Ответ теперь верный.

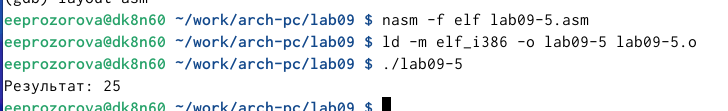


Рис. 43: Проверка программы

# 4 Выводы

В ходе лабораторной работы мной были приобретены навыки написания программ с использованием подпрограмм. Я также изучила метод отладки при помощи GDB и его основными возможностями.