Отчёт по лабораторной работе

Дисциплина: Архитектура компьютера

Кижваткина Анна Юрьевна

Содержание

# 1 Цель работы

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

# 2 Задание

1. Реализация подпрограмм в NASM.
2. Отладка программ с помощью GDB. 2.1. Добавление точек останова. 2.2. Работа с данными программы в GDB. 2.3. Обработка аргументов командной строки в GDB.
3. Выполнение самостоятельной работы.

# 3 Выполнение лабораторной работы

Создаем каталог. (рис. 1).

Рис. 1: Создание каталога

Рис. 1: Создание каталога

Перемещаемся в созданный каталог. (рис. 2).

Рис. 2: Перемещение в каталог

Рис. 2: Перемещение в каталог

Создаем файл и проверяем его наличие. (рис. 3).

Рис. 3: Создание файла и проверка наличия

Рис. 3: Создание файла и проверка наличия

Вводим текст программы листинга. (рис. 4).

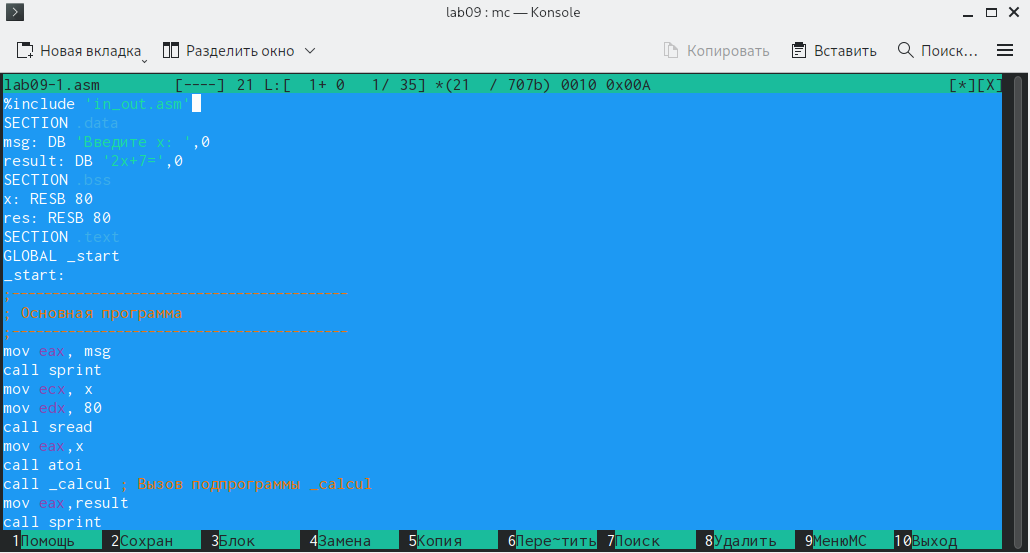


Рис. 4: Ввод программы

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

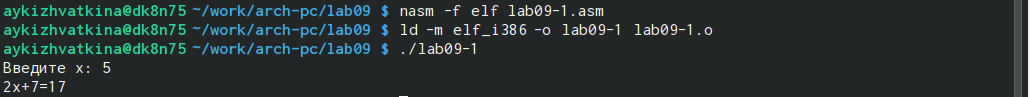


Рис. 5: Создание исполняемого файла и его запуск

Изменяем текст программы так, чтобы она решала f(g(x)). (рис. 6).

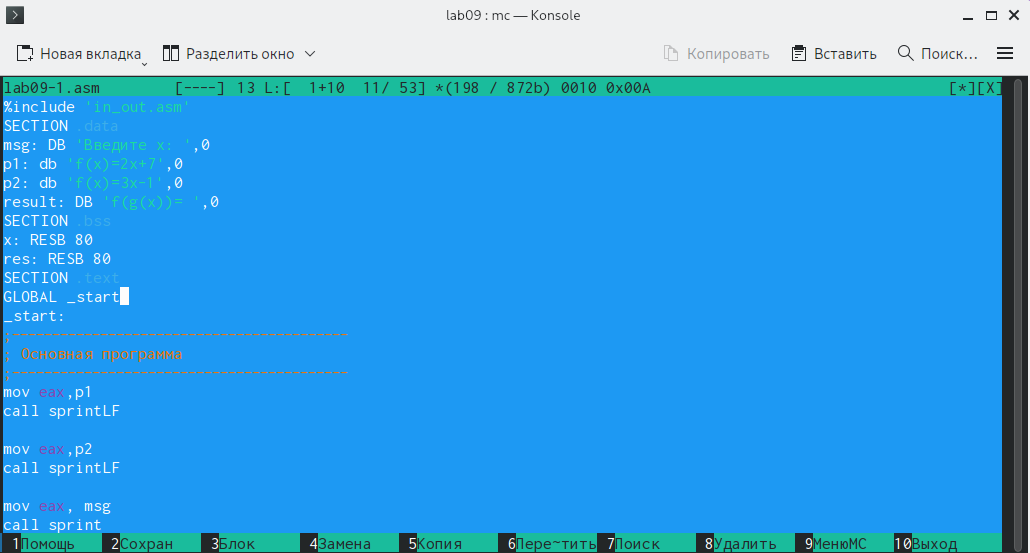


Рис. 6: Изменение программы

Создаем исполняемый файл и запускаем его. (рис. 7).

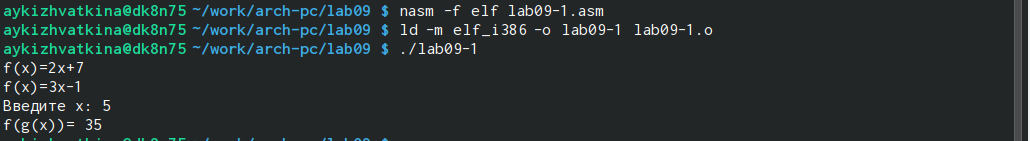


Рис. 7: Создание исполняемого файла и его запуск

Создаем файл и проверяем его наличие. (рис. 8).

Рис. 8: Создание файла и проверка наличия

Рис. 8: Создание файла и проверка наличия

Вписываем в файл программу. (рис. 9).

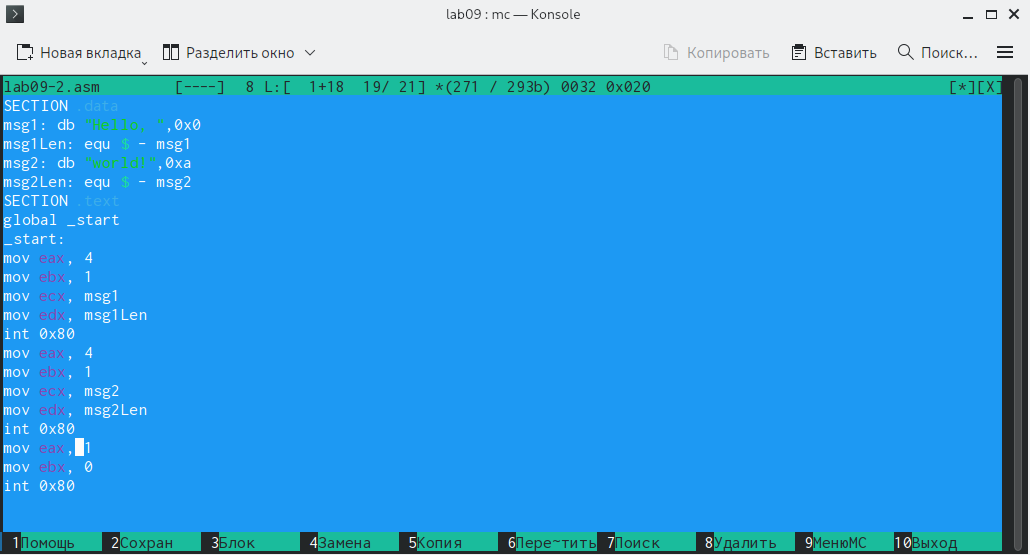


Рис. 9: Запись программы

Создаем исполняемый файл. (рис. 10).

Рис. 10: Создание исполняемого файла

Рис. 10: Создание исполняемого файла

Запуск программы в отладчик gdb. (рис. 11 и рис. 12).

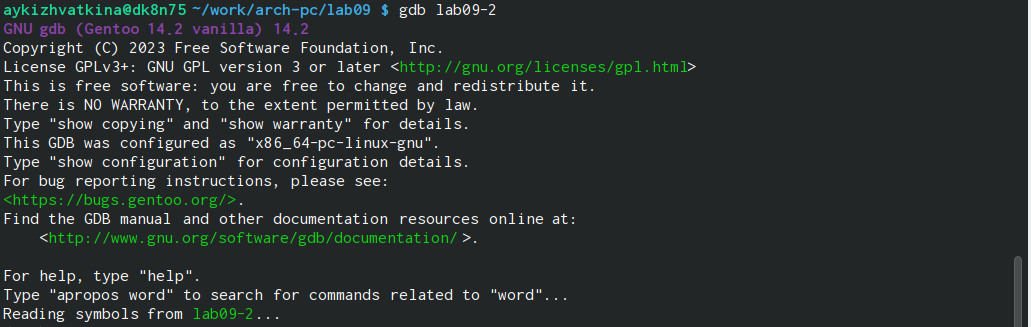


Рис. 11: Запуск программы в отладчик

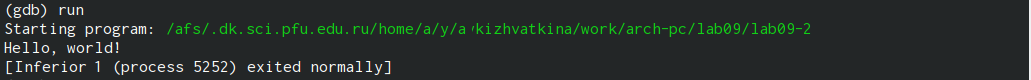


Рис. 12: Команда run

Ставим брейк на \_start. (рис. 13).

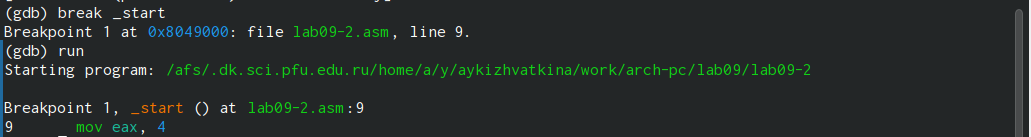


Рис. 13: Создание брейка

Смотрим код программы начиная с \_start. (рис. 14).

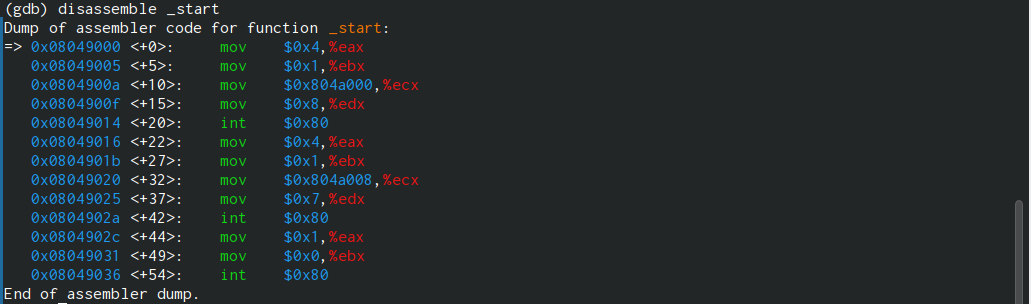


Рис. 14: Команда disassemble \_start

Переключаемся на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel. В режиме АТТ отображаются знаки % и $, а в intel нет. (рис. 15).

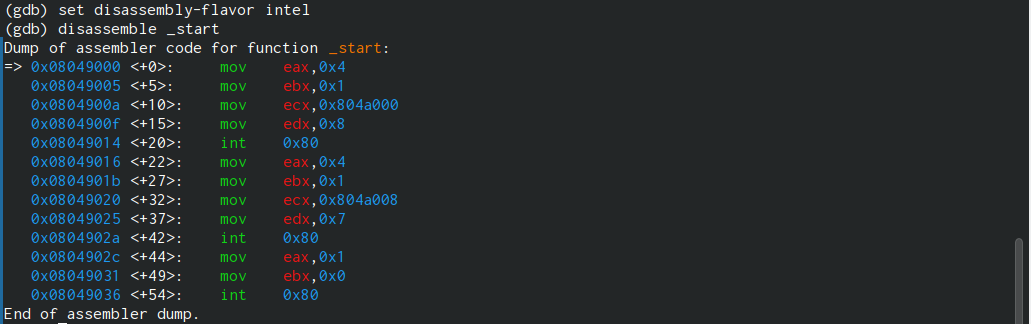


Рис. 15: Команда disassembly-flavor intel

Включаем режим псевдографики. (рис. 16).

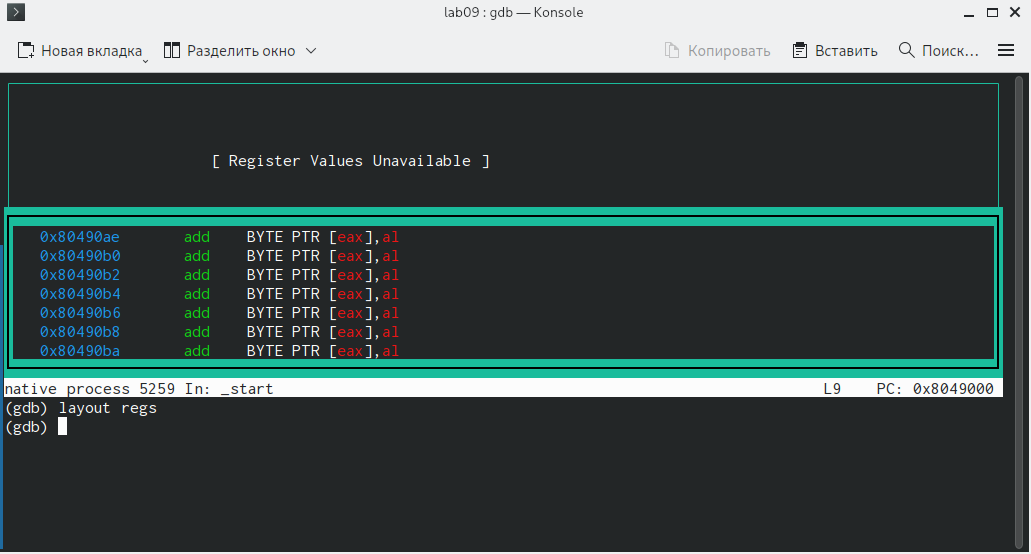


Рис. 16: Режим псевдографики

Проверяем наличие меток. (рис. 17).

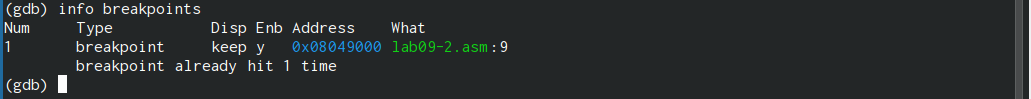


Рис. 17: Проверка меток

Добавляем метку. (рис. 18).

Рис. 18: Добавление метки

Рис. 18: Добавление метки

Проверяем наличие метки. (рис. 19).

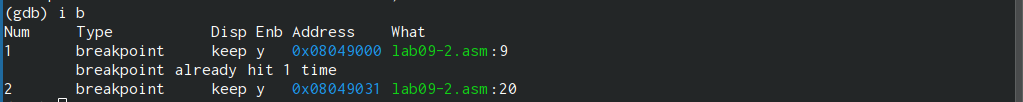


Рис. 19: Проверка меток

С помощью команды si смотрим регистры и изменяем их. (рис. 20).

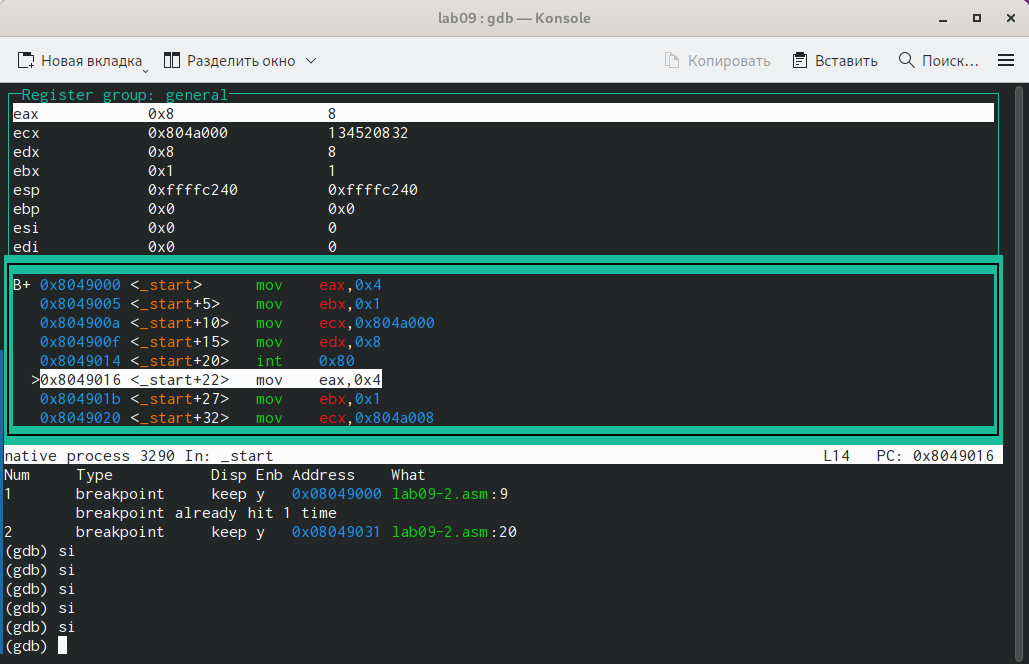


Рис. 20: Команда si

Вводим комнаду i r.

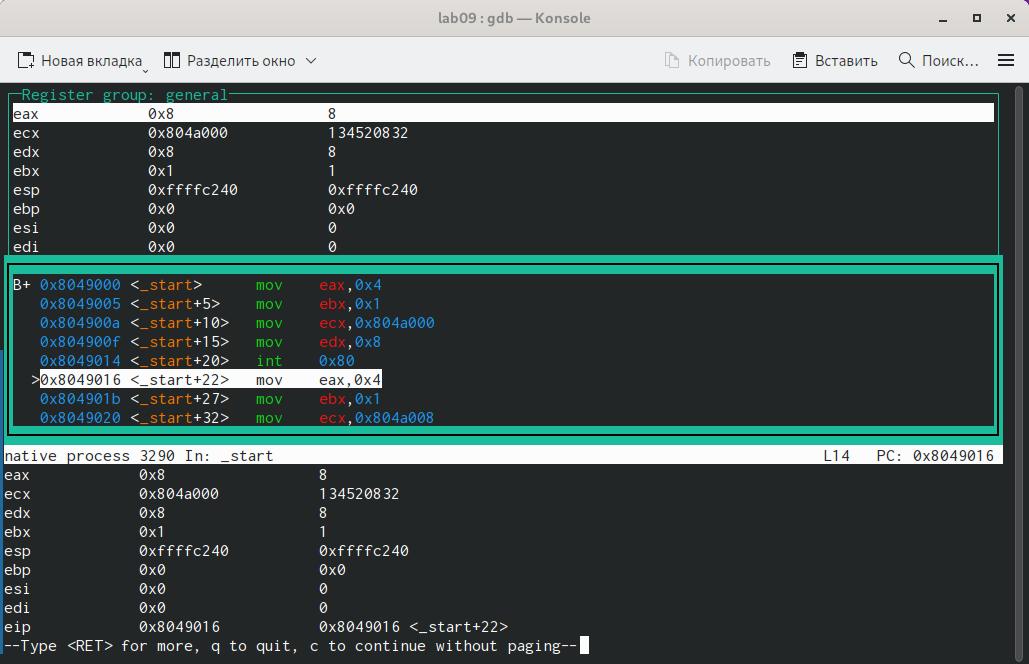


Рис. 21: Команда i r

С помощью команды смотрим значение переменной msg1. (рис. 22).

Рис. 22: Просмотр значения переменной

Рис. 22: Просмотр значения переменной

Смотрим значение переменной msg2. (рис. 23).

Рис. 23: Просмотр значения переменной

Рис. 23: Просмотр значения переменной

С помощью команды set меняем значение переменной msg1. (рис. 24).

Рис. 24: Команда set

Рис. 24: Команда set

С помощью команды set меняем значение переменной msg2. (рис. 25).

Рис. 25: Изменение msg2

Рис. 25: Изменение msg2

Выводим значение регистров ecx и eax. (рис. 26).



Рис. 26: Вывод значений

Изменяем значение регистра ebx. Команда выводит два разных значения так как в первый раз мы вносим значение 2, а во второй раз регистр равен двум. (рис. 27).

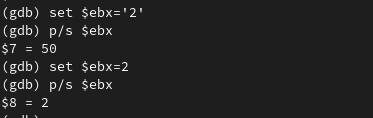


Рис. 27: Изменение значений регистра

Копируем файл lab8-2.asm в папку. (рис. 28).

Рис. 28: Команда cp

Рис. 28: Команда cp

Создаем исполняемый файл. (рис. 29).

Рис. 29: Создание исполняемого файла

Рис. 29: Создание исполняемого файла

Запускаем файл в отладчике с указанием аргументов. (рис. 30).

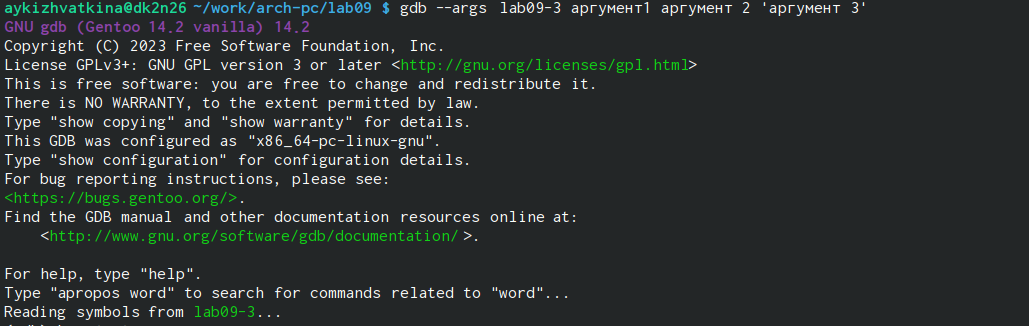


Рис. 30: Запуск файла

Ставим метку на \_start. (рис. 31).

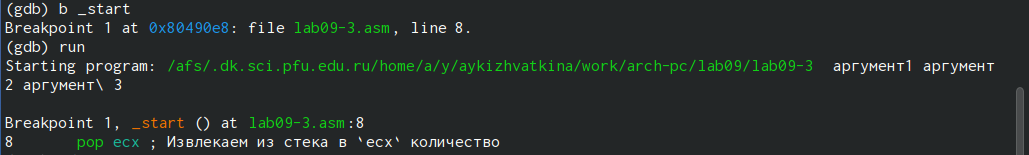


Рис. 31: Создание метки

Проверяем количество элементов в вершине стека. (рис. 32).

Рис. 32: Проверка количества элементов

Рис. 32: Проверка количества элементов

Просматриваем позиции стека. Элементы расположены с интервалом в 4 единицы, так как стек может хранить до 4 байт.

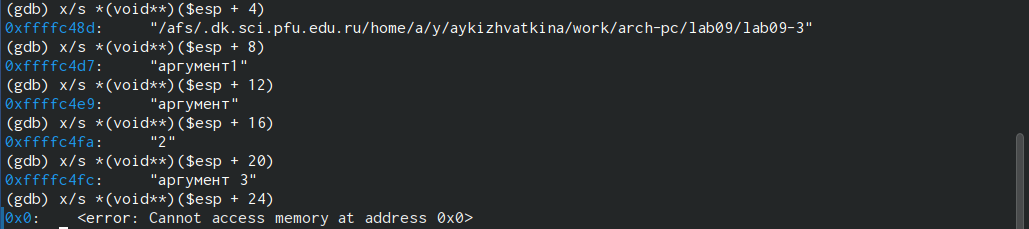


Рис. 33: Проверка позиций стека

Переделываем программу из лабораторной работы 8 так, чтобы вычисления были подпрограммой. (рис. 34).

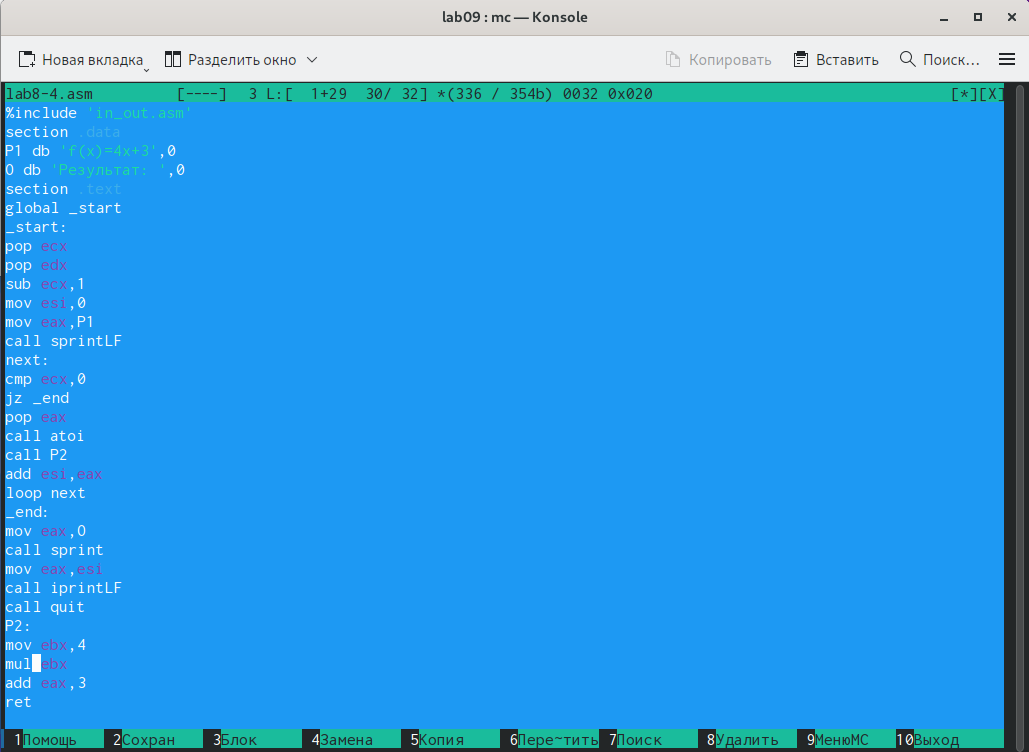


Рис. 34: Изменение программы

Создаем исполняемый файл. (рис. 35).

Рис. 35: Создание исполняемого файла

Рис. 35: Создание исполняемого файла

Запускаем файл и проверяем правильность выполнения. (рис. 36).

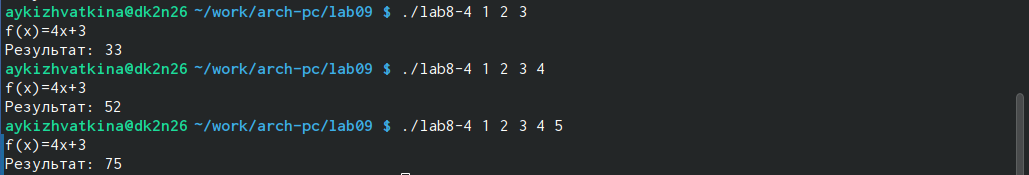


Рис. 36: Запуск файла

Создаем файл. (рис. 37).

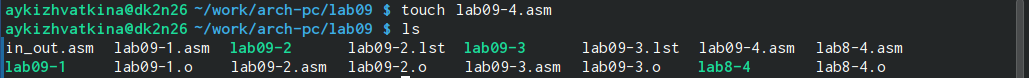


Рис. 37: Создание файла

Переписываем программу в файл. (рис. 38).

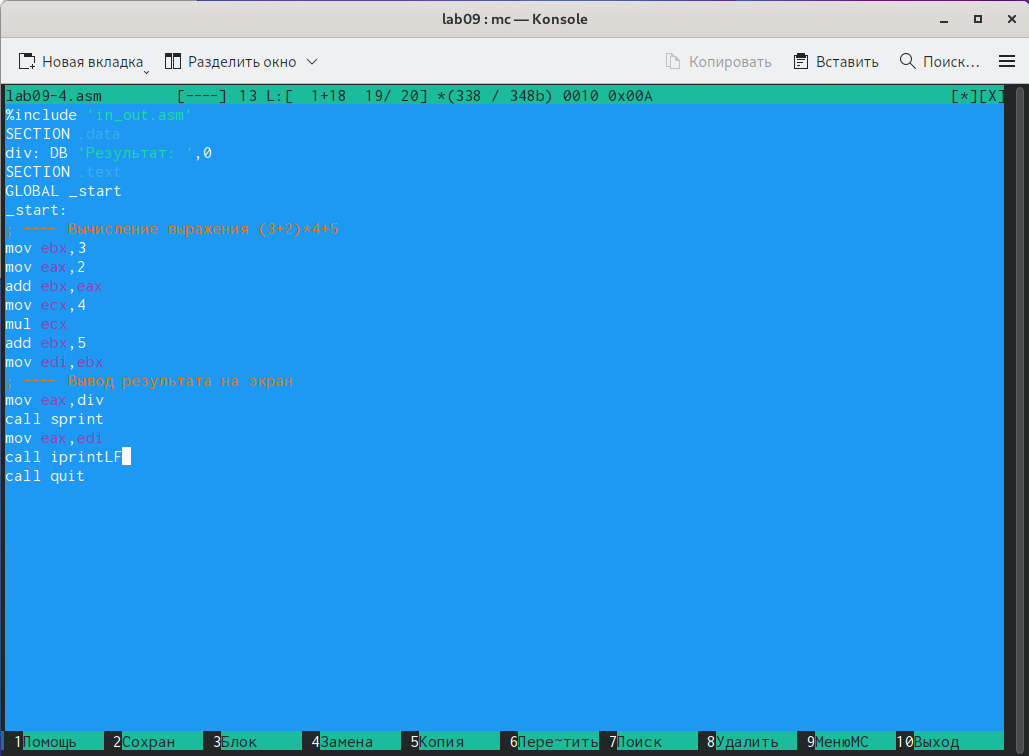


Рис. 38: Программа в файле

Создаем исполняемый файл и запускаем его. (рис. 39). Программа выводит неверный ответ.

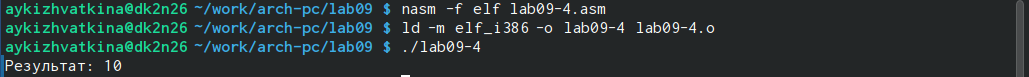


Рис. 39: Создание исполняемого файла и запуск

Запускаем программу в отладчике. (рис. 40).

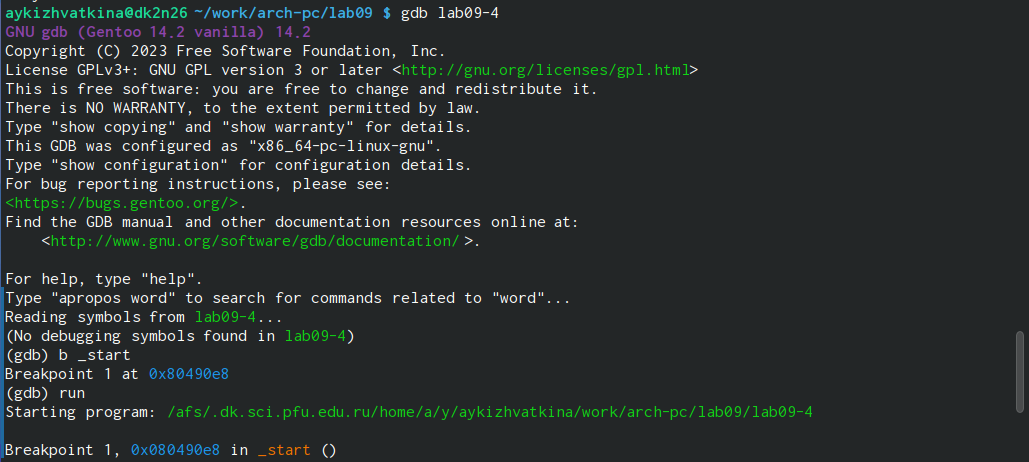


Рис. 40: Запуск программы в отладчике

Открываем регистры и анализируем их. Я поняла, что некоторые регистры стоят не на своих местах. Исправляем это. (рис. 41).

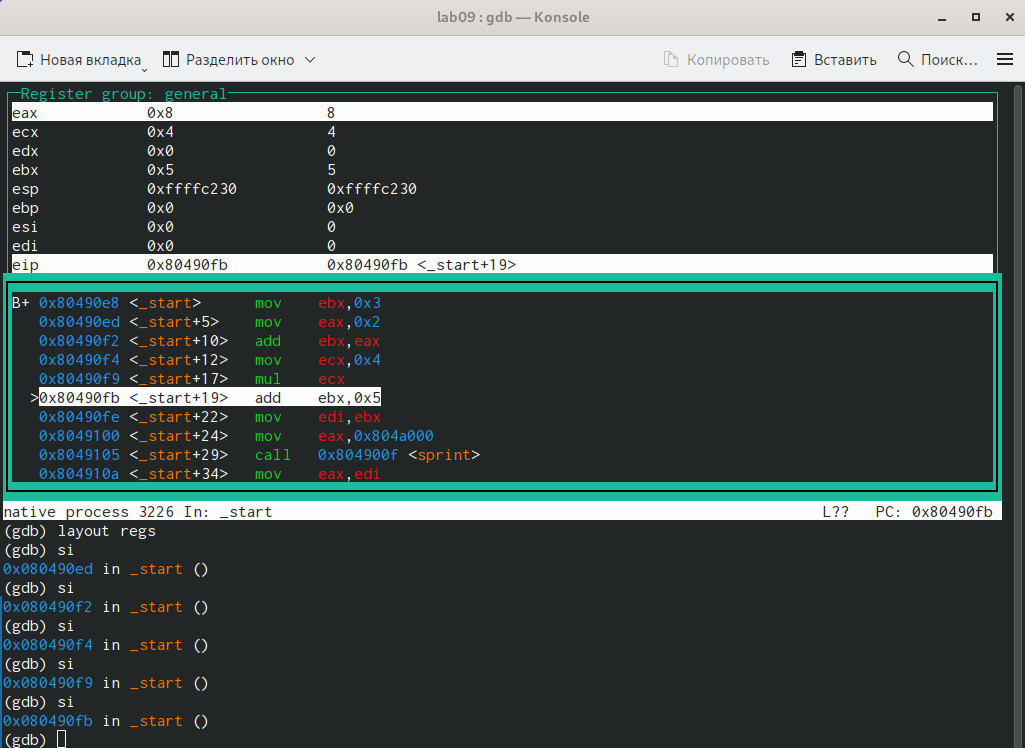


Рис. 41: Анализ регистров

Создаем исполняемый файл и запускаем его. Проверяем правильность выполнения. (рис. 42).

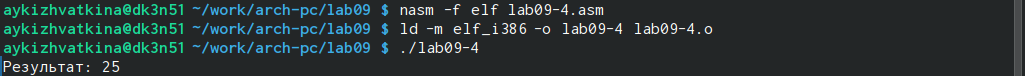


Рис. 42: Создание исполняемого файла и запуск

# 4 Выводы

С помощью данной лабораторной работы мы приобрели навыки написания программ с использованием подпрограмм. Ознакомились с методами отладки при помощи GDB и его основными возможностями.