Отчет по лабораторной работе №10

Дисциплина: Архитектура компьютеров

Лобанова Полина Иннокентьевна

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. Создадим каталог lab10 и текстовый файл lab10-1.asm.

Рис. 1: Создание каталога lab10 и файла lab10-1.asm.

Рис. 1: *Создание каталога lab10 и файла lab10-1.asm.*

1. Заполним текстовый файл lab10-1.asm.

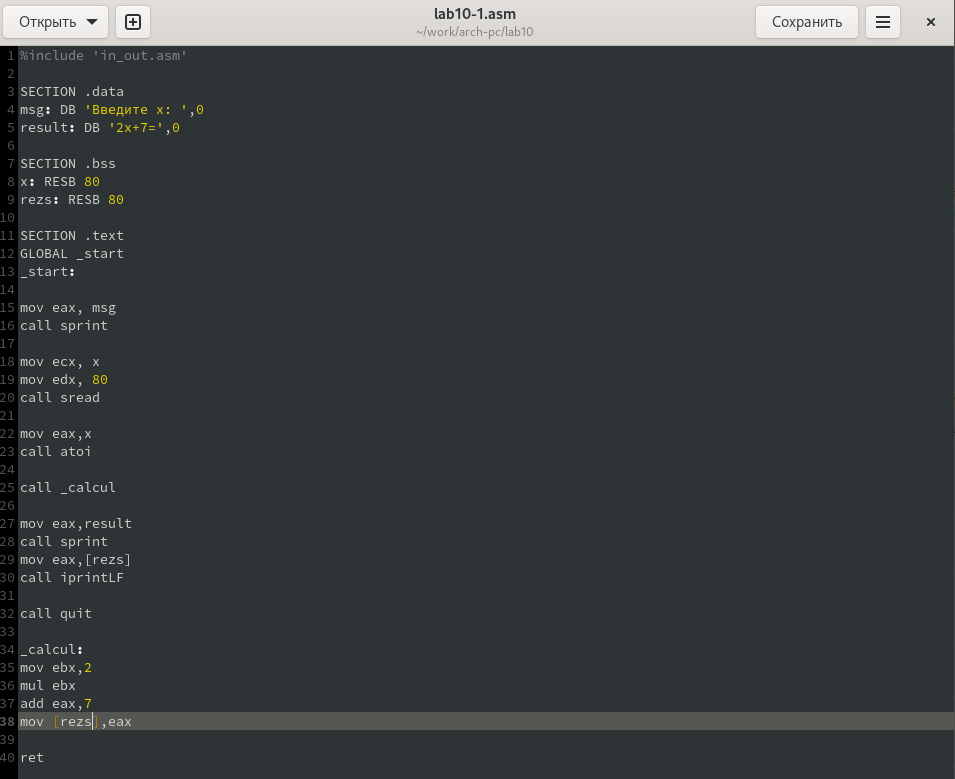


Рис. 2: *Текст программы в файле lab10-1.asm.*

1. Создадим исполняемый файл и проверим его работу.

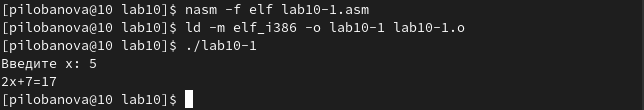


Рис. 3: *Создание исполняемого файла и его запуск.*

1. Изменим текст программы, добавив подпрограмму \_subcalcul в подпрограмму \_calcul, для вычисления выражения f(g(x)), где x вводится с клавиатуры, f(x) = 2x + 7, g(x) = 3x − 1.

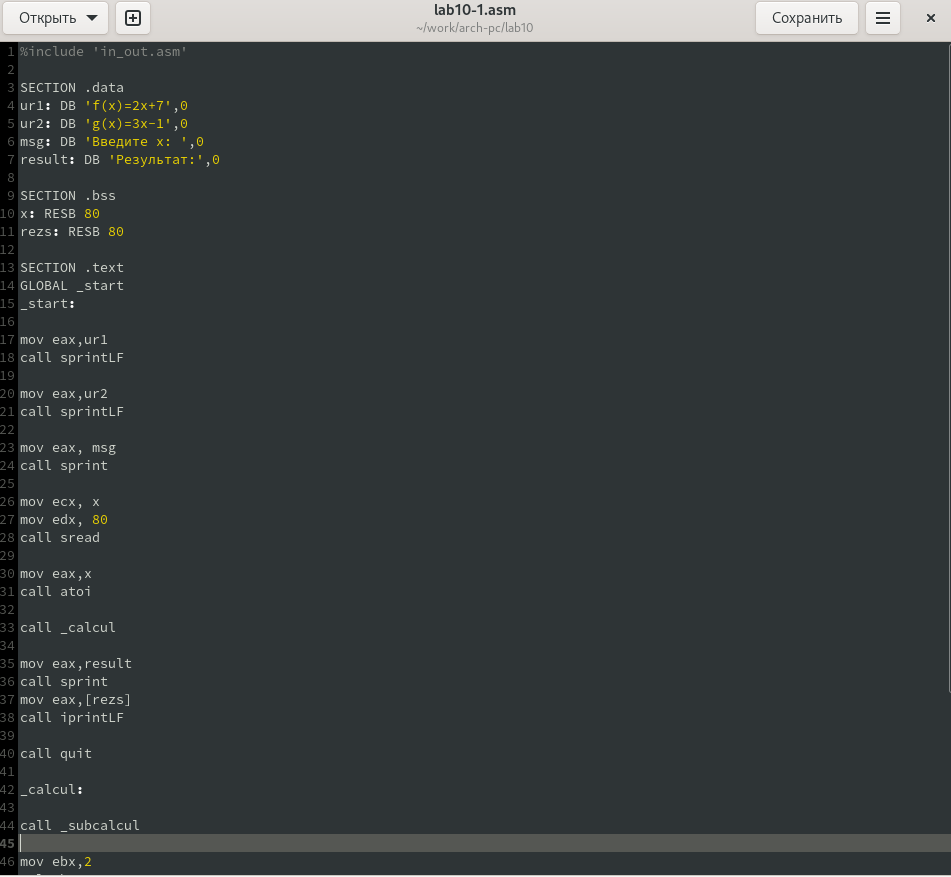


Рис. 4: *Измененный текст программы в файле lab10-1.asm.*

1. Создадим исполняемый файл и проверим его работу.

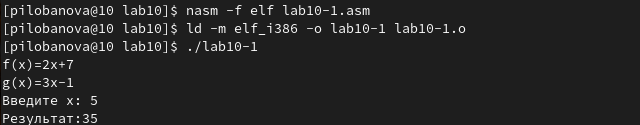


Рис. 5: *Создание исполняемого файла и его запуск.*

1. Создадим текстовый файл lab10-2.asm.

Рис. 6: Создание текстового файла lab10-2.asm.

Рис. 6: *Создание текстового файла lab10-2.asm.*

1. Заполним файл lab10-2.asm.

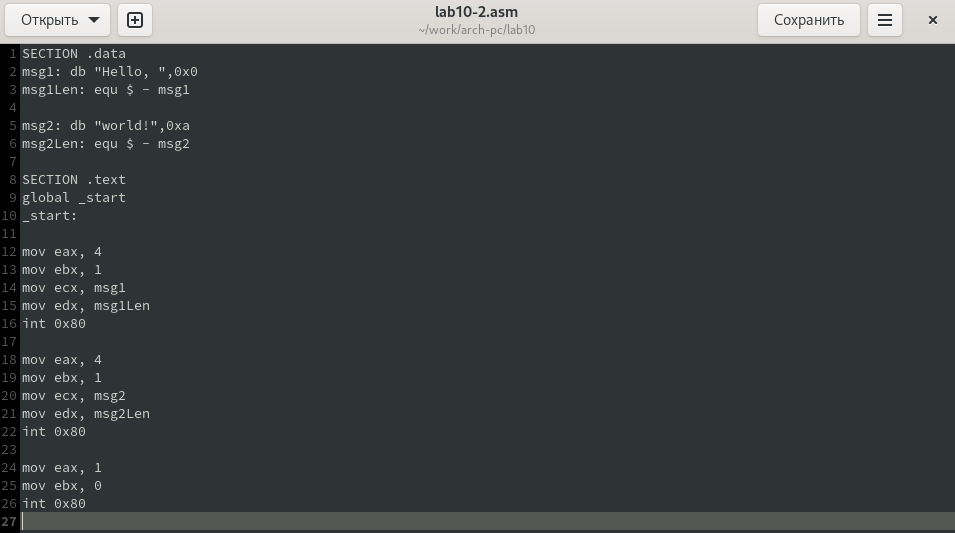


Рис. 7: *Текст программы в файле lab10-2.asm.*

1. Создадим исполняемый фай и проверим его работу.

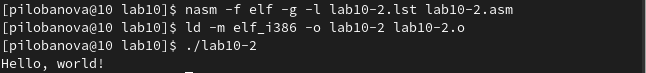


Рис. 8: *Создание исполняемого файла и его запуск.*

1. Загрузим исполняемый файл в отладчик gdb.

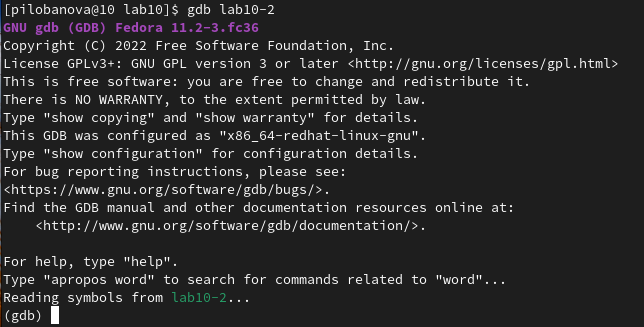


Рис. 9: *Загрузка исполняемого файла на gdb.*

1. Проверим работу программы, запустив ее в оболочке GDB с помощью команды run.

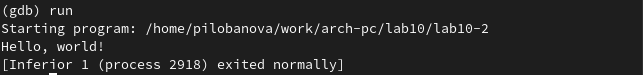


Рис. 10: *Запуск программы.*

1. Установим брейкпоинт на метку \_start и запустим.

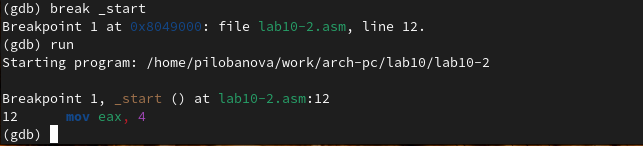


Рис. 11: *Установка брейкпоинта и запуск программы.*

1. Посмотрим дисассимилированный код программы с помощью команды disassemble начиная с метки \_start.

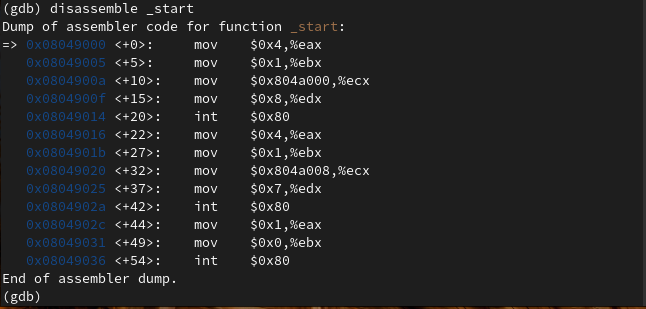


Рис. 12: *Дисассимилированный код.*

1. Переключимся на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel.

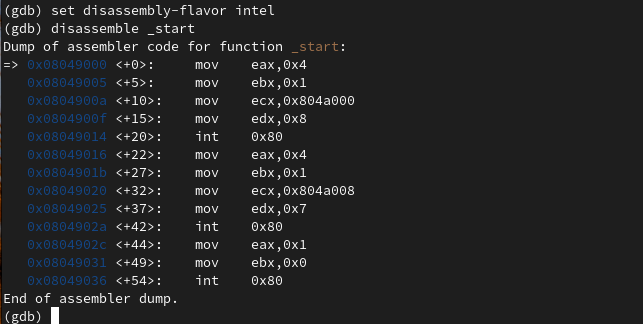


Рис. 13: *Переход на отображение команд с Intel’овским синтаксисом.*

Различия отображения синтаксиса машинных команд в режимах ATT и Intel заключаются в том, что у ATT регистры стоят справа и перед каждым регистром стоит символ %, также перед каждой строкой стоит символ $, у Intel же регистры стоят слева и нет дополнительных символов.

1. Включим режим псевдографики.

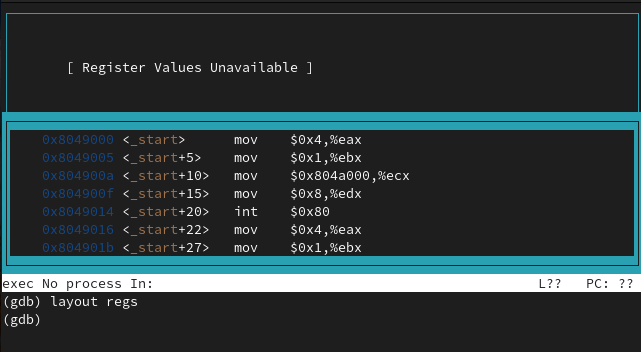


Рис. 14: *Режим псевдографики.*

1. Проверим наличие брейкпоинтов.

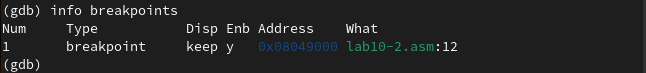


Рис. 15: *Проверка брейкпоинтов.*

1. Установим еще одну точку останова по адресу инструкции.

Рис. 16: Установка точки останова.

Рис. 16: *Установка точки останова.*

1. Посмотрим информацию о всех установленных точках останова.

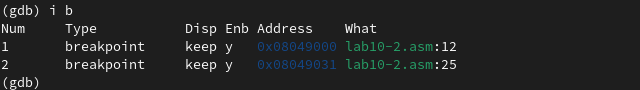


Рис. 17: *Провека точек останова.*

1. Выполним 5 инструкций с помощью команды stepi.



Рис. 18: *Значения регистров до команды stepi.*

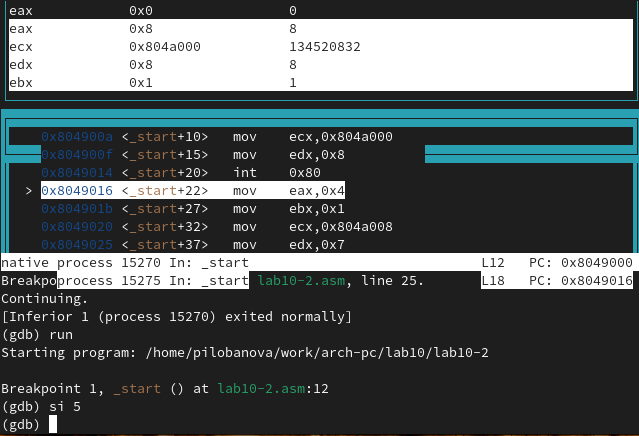


Рис. 19: *Значения регистров после команды stepi.*

Значение регистра eax изменилось с 0 на 8. Значение регистра ecx изменилось с 0 на 134520832. Значение регистра edx изменилось с 0 на 8. И значение регистра ebx изменилось с 0 на 1.

1. Посмотрим содержимое регистров с помощью команды info registers.

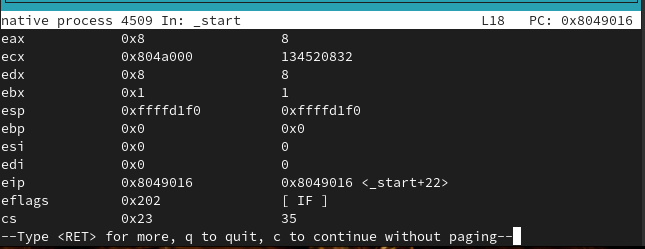


Рис. 20: *Содержимое регистров.*

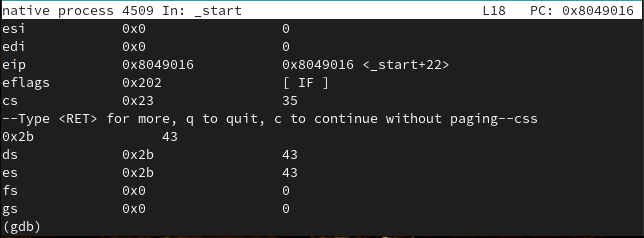


Рис. 21: *Содержимое регистров.*

1. Посмотрим значение переменной msg1 по имени.

Рис. 22: Значение переменной msg1.

Рис. 22: *Значение переменной msg1.*

1. Посмотрим значение переменной msg2 по адресу.

Рис. 23: Значение переменной msg2.

Рис. 23: *Значение переменной msg2.*

1. Изменим первый символ переменной msg1.



Рис. 24: *Изменение переменной msg1.*

1. Заменим символ в переменной msg2.

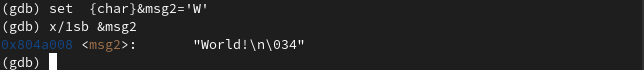


Рис. 25: *Изменение переменной msg2.*

1. Выведем в различных форматах значение регистра edx.



Рис. 26: *Значение регистра edx в различных форматах.*

1. Изменим значение регистра edx с помощью команды set.



Рис. 27: *Изменение значение регистра edx.*

Разница вывода команд p/s $ebx заключается в том, что в первом случае мы вносим значение 2, а во втором регистр приравнивается к 2.

1. Завершим выполнение программы с помощью команды continue.

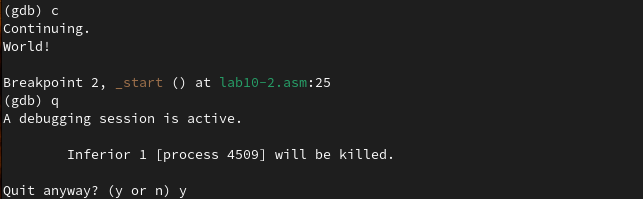


Рис. 28: *Завершение выполнения программы.*

1. Скопируем файл lab9-2.asm в каталог lab10 с названием lab10-3.asm.

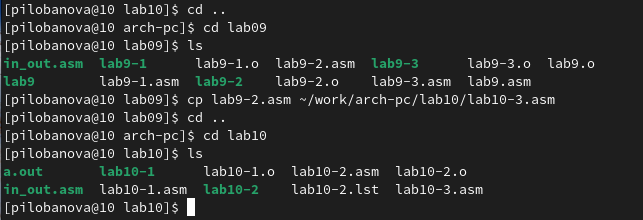


Рис. 29: *Копирование файла lab9-2.asm.*

1. Создадим исполняемый файл.

Рис. 30: Создание исполняемого файла.

Рис. 30: *Создание исполняемого файла.*

1. Загрузим исполняемый файл в отладчик, указав аргументы.

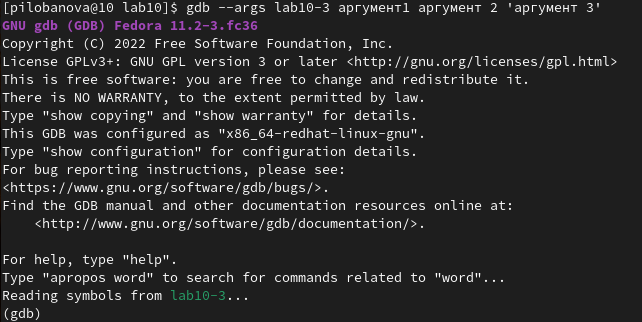


Рис. 31: *Загрузка исполняемого файла в отладчик.*

1. Установим точку останова перед первой инструкцией в программе и запустим ее.

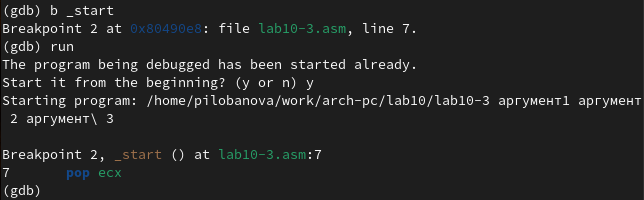


Рис. 32: *Установка точки останова.*

1. Посмотрим число аргументов командной строки.

Рис. 33: Число аргументов командной строки.

Рис. 33: *Число аргументов командной строки.*

1. Посмотрим остальные позиции стека.

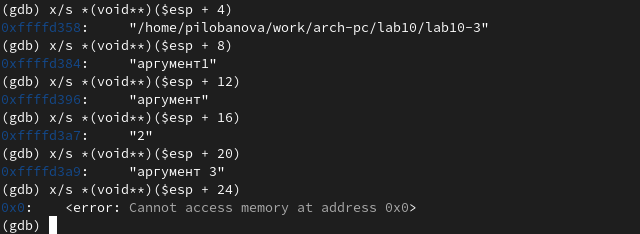


Рис. 34: *Позиции стека.*

Шаг изменения адреса равен 4, потому что стек может хранить до 4 байт.

# 3 Выполнение самостоятельной работы

1. Перенесем файл lab9.asm в каталог lab10.

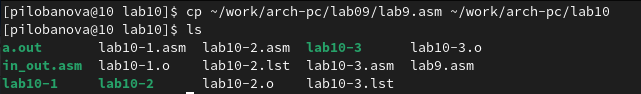


Рис. 35: *Копирование файла lab9.asm в каталог lab10.*

1. Преобразуем программу из лабораторной работы №9, реализовав вычисление значения функции f(x), как подпрограмму.

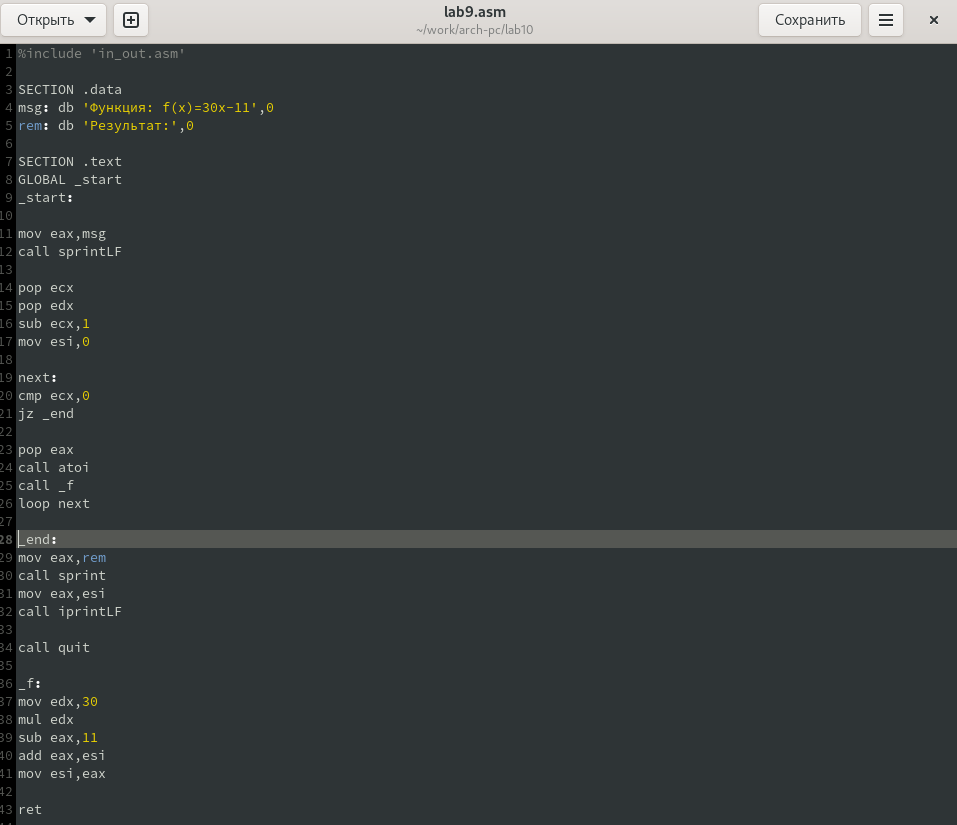


Рис. 36: *Измененный текст программы в файле lab9.asm.*

1. Создадим исполняемый файл и проверим его работу.

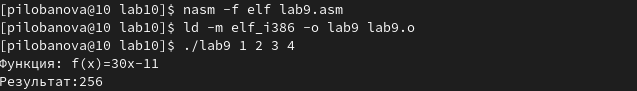


Рис. 37: *Создание исполняемого файла и его запуск.*

1. Создадим текстовый файл lab10.asm.

Рис. 38: Создание текстового файла lab10.asm.

Рис. 38: *Создание текстового файла lab10.asm.*

1. Запишем программу вычисления выражения (3 + 2) \* 4 + 5.

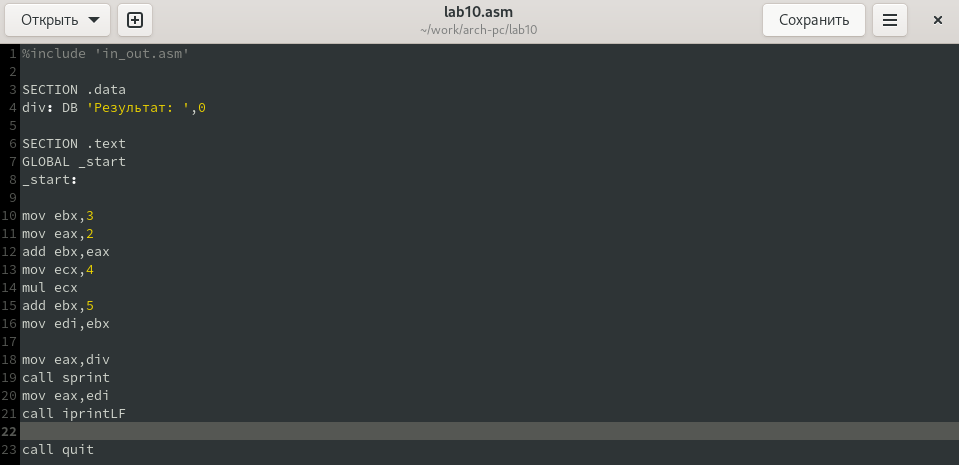


Рис. 39: *Текст программы в файле lab10.asm.*

1. Создадим исполняемый файл и проверим его работу. Как и ожидалось, программа выдает неверный результат.

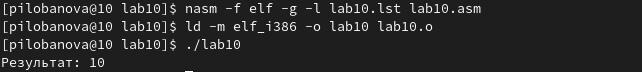


Рис. 40: *Создание исполняемого файла и его запуск.*

1. Используем отладчик для определения ошибки.

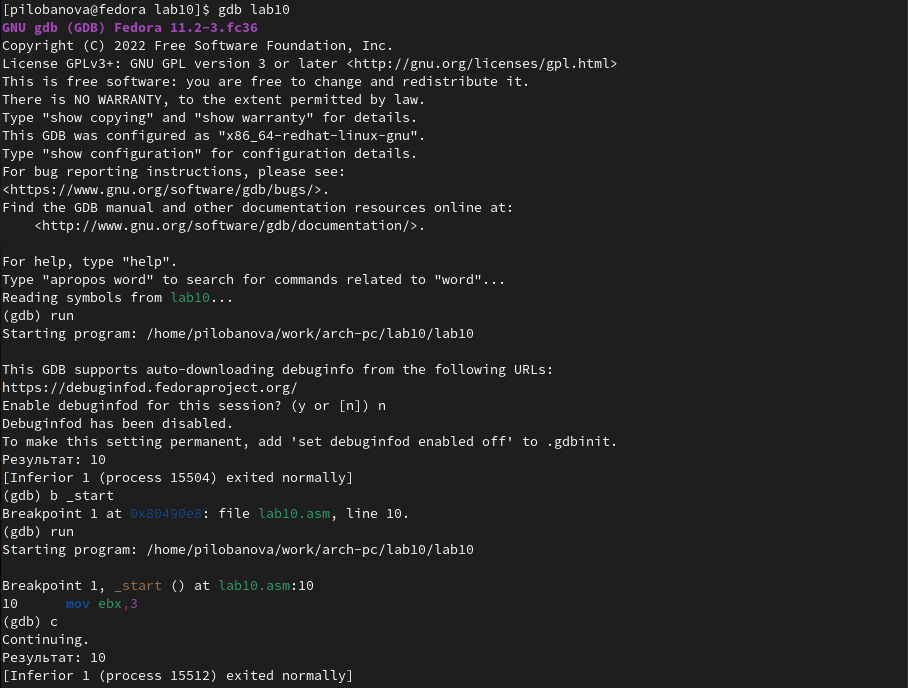


Рис. 41: *Использование отладчика.*



Рис. 42: *Использование отладчика.*

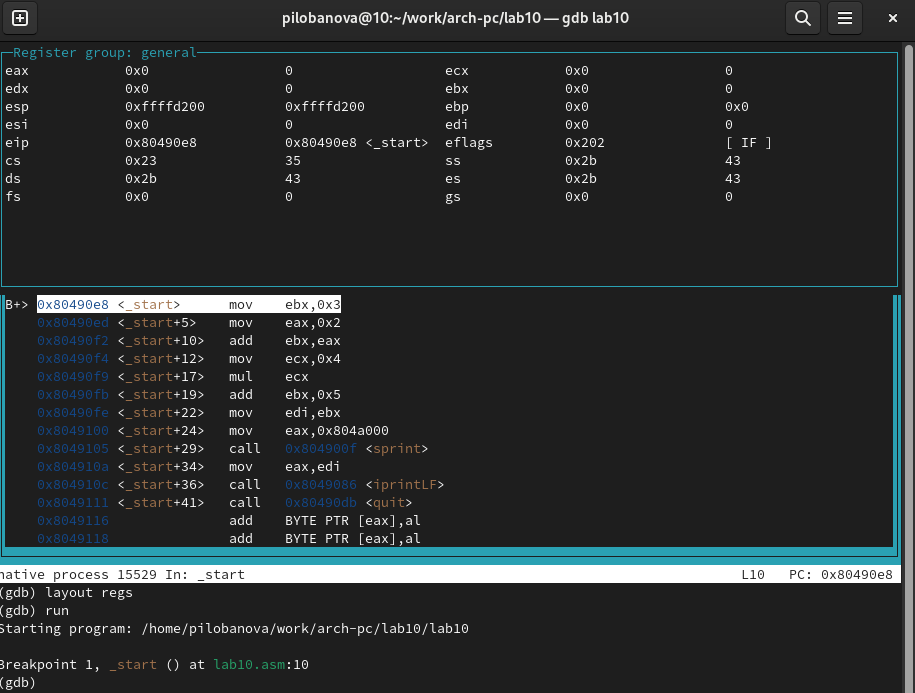


Рис. 43: *Использование отладчика.*

1. Пошагово проверим каждое действие и проследим изменения регистров.

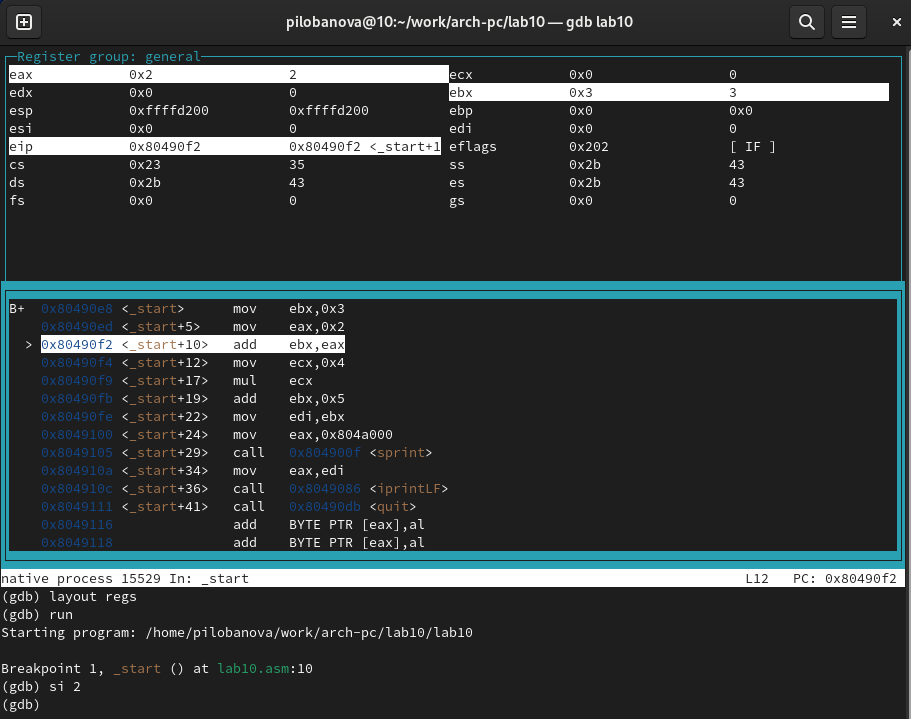
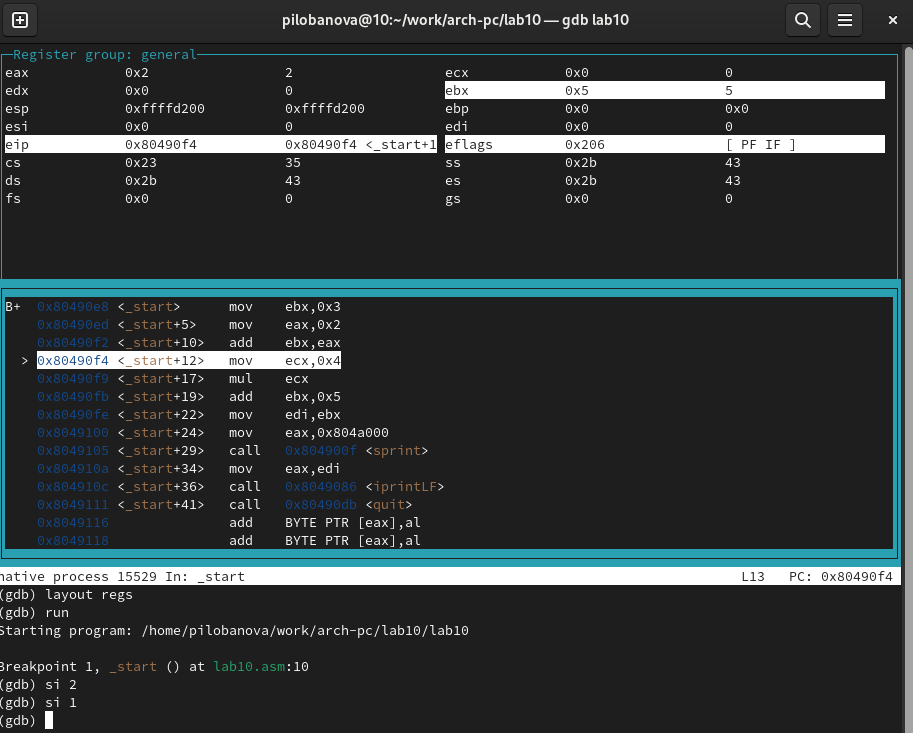
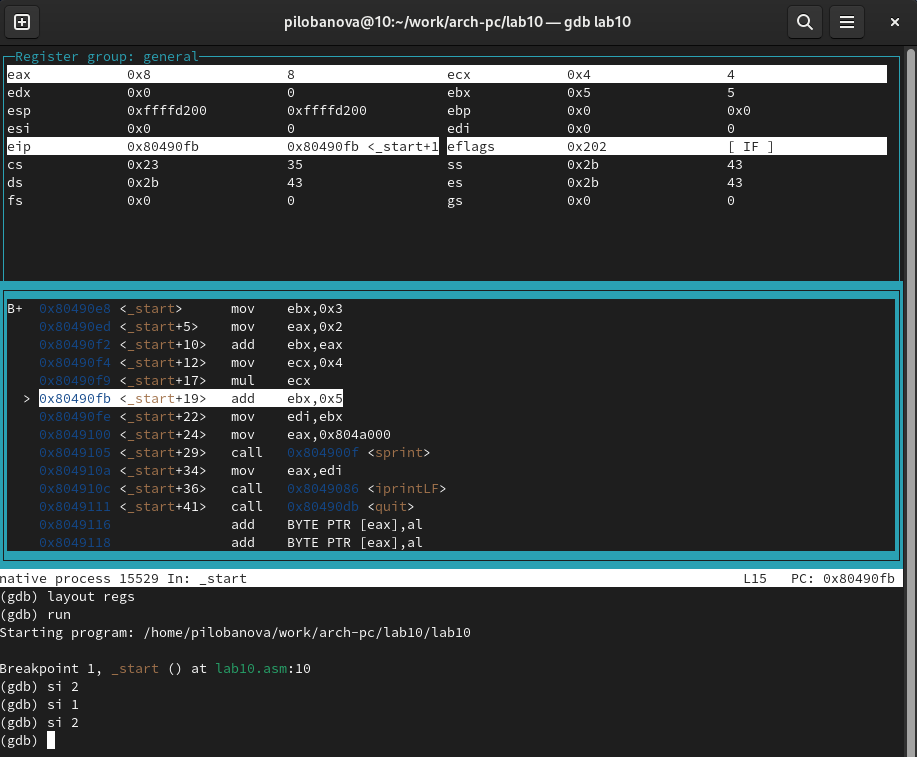


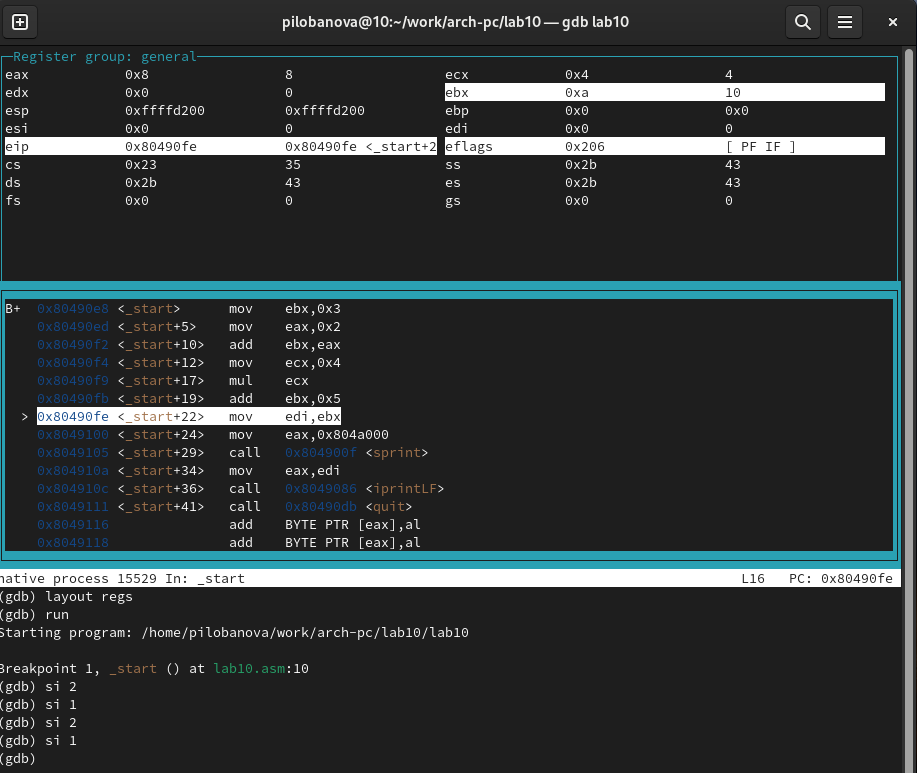
Рис. 44: *Первые две инструкции.*

44

Из-за того, что при сложении не соблюден необходимый порядок регистров, результат сложения записывается в регистр ebx.

45

Поскольку результат сложения записан не в eax, умножение выполняется неверно.

46

Из-за того, что умножение выполнено неверно, прибавление 5 к регистру ebx выдает не тот ответ, который должен быть.

1. Исправим ошибки в тексте программы.

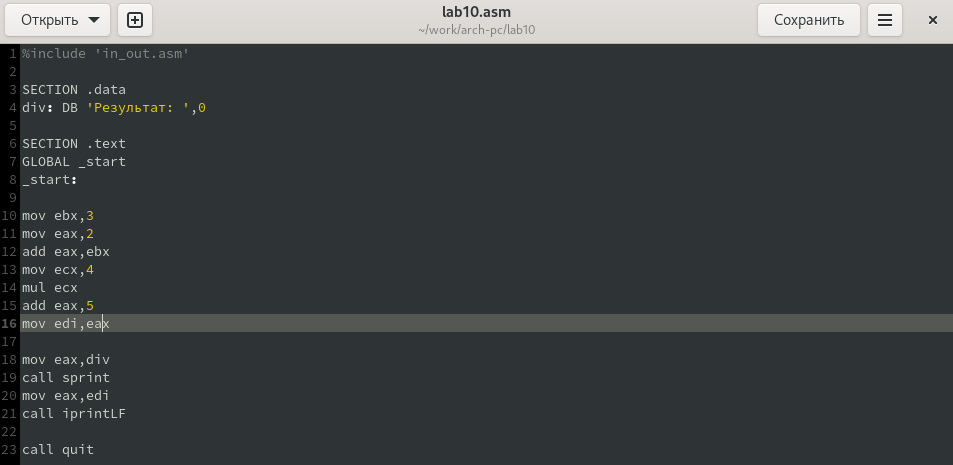


Рис. 45: *Измененный текст программы в файле.*

1. Создадим исполняемый файл и проверим его работу. После изменений программа начала работать исправно.

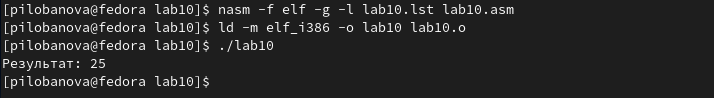


Рис. 46: *Создание исполняемого файла и его запуск.*

# 4 Вывод

Я научилась писать программы с использованием подпрограмм, а также ознакомилась с методами отладки и при помощи GDB и его основными возможностями.