ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №10

Дисциплина: Архитектура компьютера

Обрезкова Анастасия Владимировна

Содержание

# 1 Цель работы

Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

# 2 Задание

1. Приобрести навыки написания программ с использованием подпрограмм.
2. Ознакомится с методами отладки при помощи GDB и его основными возможностями.

# 3 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа:

• обнаружение ошибки;

• поиск её местонахождения;

• определение причины ошибки;

• исправление ошибки.

Наиболее часто применяют следующие методы отладки:

• создание точек контроля значений на входе и выходе участка программы (например, вывод промежуточных значений на экран — так называемые диагностические сообщения);

• использование специальных программ-отладчиков.

Отладчики позволяют управлять ходом выполнения программы, контролировать и изменять данные. Это помогает быстрее найти место ошибки в программе и ускорить её исправление. Наиболее популярные способы работы с отладчиком — это использование точек останова и выполнение программы по шагам.

GDB (GNU Debugger — отладчик проекта GNU) [1] работает на многих UNIX-подобных системах и умеет производить отладку многих языков программирования. GDB предлагает обширные средства для слежения и контроля за выполнением компьютерных программ. Отладчик не содержит собственного графического пользовательского интерфейса и использует стандартный текстовый интерфейс консоли. Однако для GDB существует несколько сторонних графических надстроек, а кроме того, некоторые интегрированные среды разработки используют его в качестве базовой подсистемы отладки.

# 4 Выполнение лабораторной работы

## 4.1 Реализация подпрограмм в NASM

1. Создала каталог для программ лабораторной работы №10, перешла в него и создала файл lab9=10-1.asm. (рис. 1)

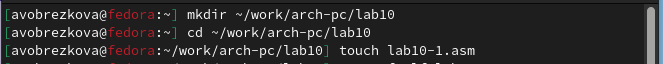


Рис. 1: Создание, переход в lab09

1. Ввела в файл lab10-1 нужный текст программы из листинга 10.1., создала исполняемый файл и вывела результат. (рис. 2; рис. 3)

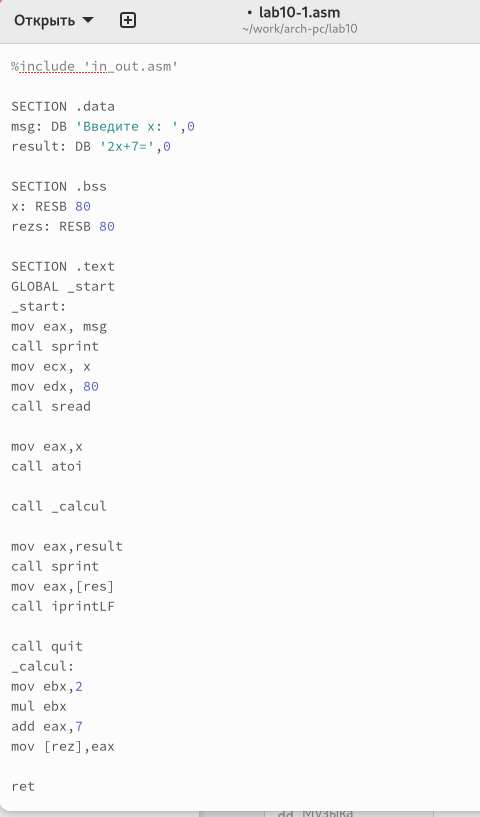


Рис. 2: Ввод текста

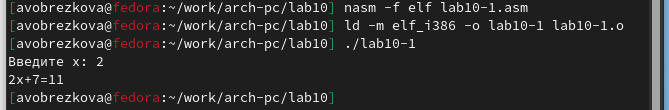


Рис. 3: Результат программы

1. Изменила текст программы, для вычисления выражения f(g(x)), где x вводится с клавиатуры, f(x)=2x+7, g(x)=3x-1. (рис. 4; рис. 5; рис. 6)

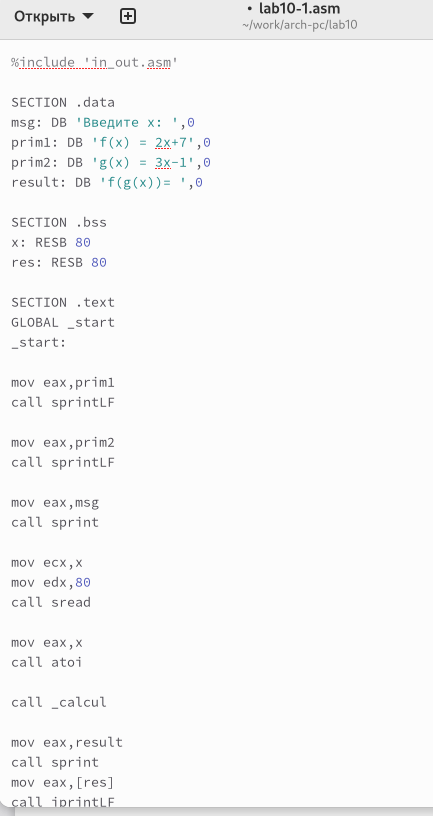


Рис. 4: Изменения текста

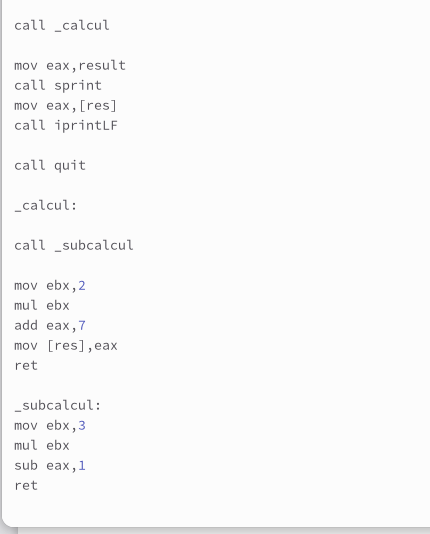


Рис. 5: Изменения текста

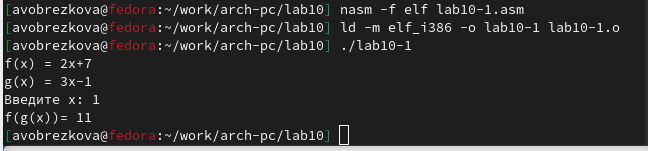


Рис. 6: Результат изменений

## 4.2 Отладка программ с помощью GDB

1. Создала файл lab10-2.asm с текстом программы из листинга 10.2. Получила исполняемый файл для работы с GDB и загрузила его в gdb.(рис. 7; рис. 8)

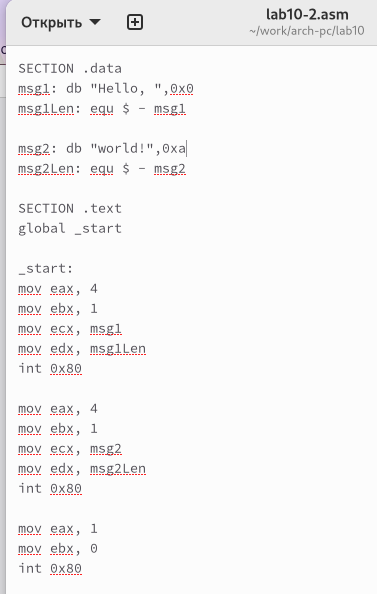


Рис. 7: Текст программы

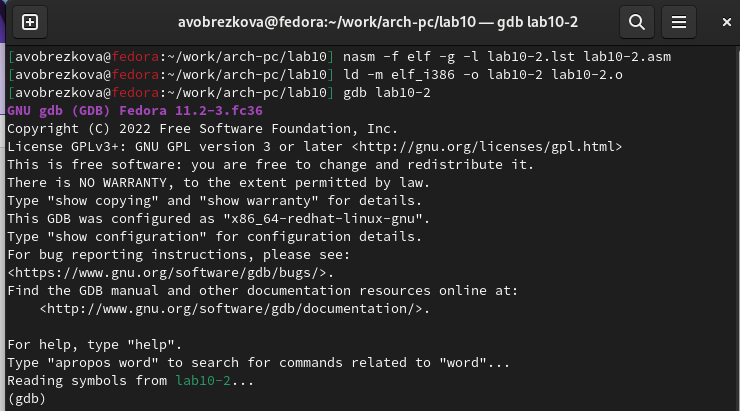


Рис. 8: Вывела результат

1. Проверьте работу программы, запустив ее в оболочке GDB с помощью команды run. (рис. 9)

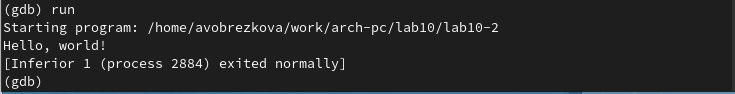


Рис. 9: Проверка работы

1. Установила брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запустила её. (рис. 10)

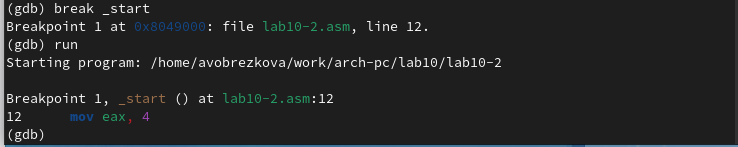


Рис. 10: Установка брейкпоинта

1. Посмотрела дисассимилированный код программы с помощью команды disassemble начиная с метки \_start. (рис. 11)

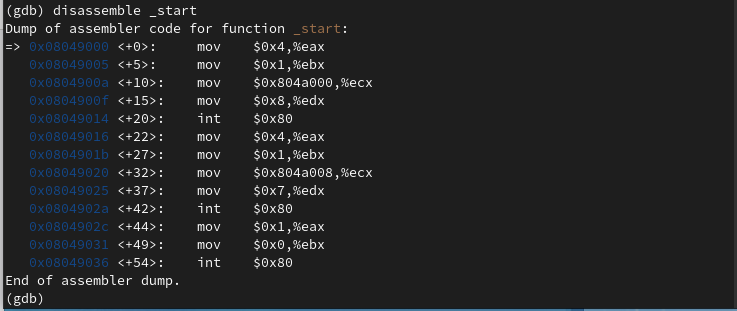


Рис. 11: Просмотр кода

1. Переключидась на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel. (рис. 12)

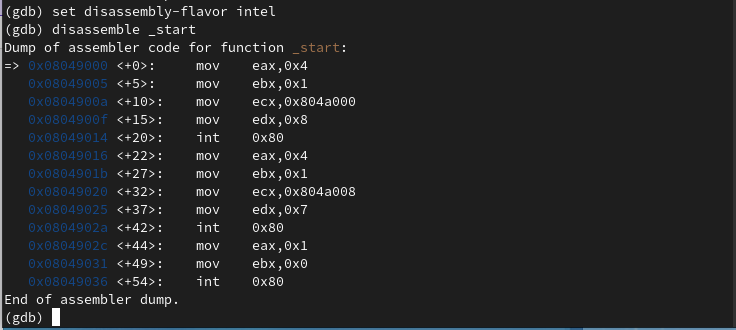


Рис. 12: Переключение на отображение

Различия отображения синтаксиса машинных команд в режимах ATT и Intel: в АТТ перед адресом регистра ставится $, а перед названием регистра %, сначала записывается адрес, а потом - регистр. В Intel сначала регистр, а потом адрес, и перед ними ничего не ставится.

1. Включите режим псевдографики для более удобного анализа программы. (рис. 13)

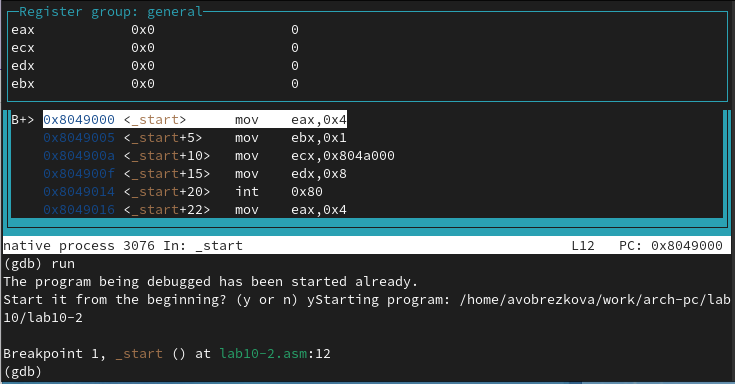


Рис. 13: Режим псевдографики

## 4.3 Добавление точек останова

1. Ранее я установила точку останова по имени метки (\_start). Проверила это с помощью команды info breakpoints (кратко i b). (рис. 14)

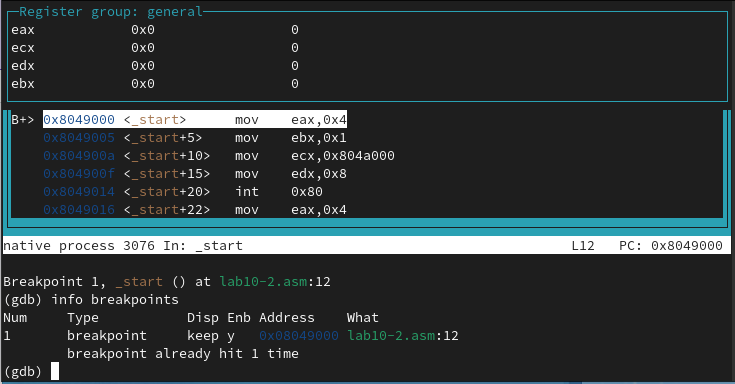


Рис. 14: Проверка

1. Установила еще одну точку останова по адресу инструкции.. (рис. 15)

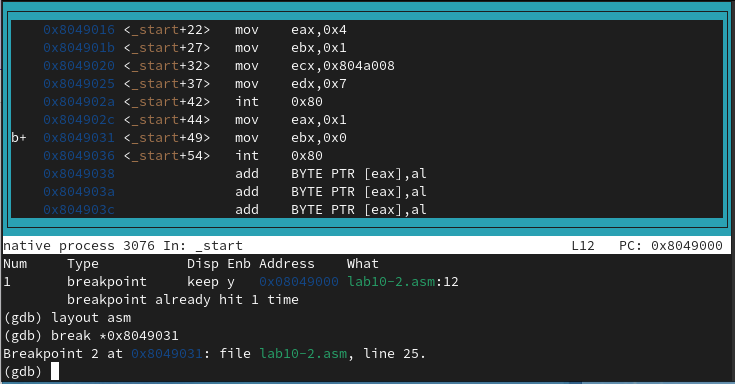


Рис. 15: Точка остановы

1. Посмотрела информацию о всех установленных точках останова. (рис. 16)

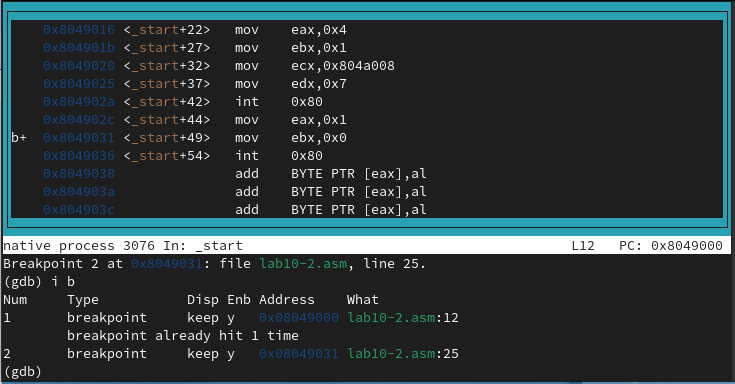


Рис. 16: Информация

## 4.4 Работа с данными программы в GDB

1. Посмотрела содержимое регистров с помощью команды info registers. (рис. 17)

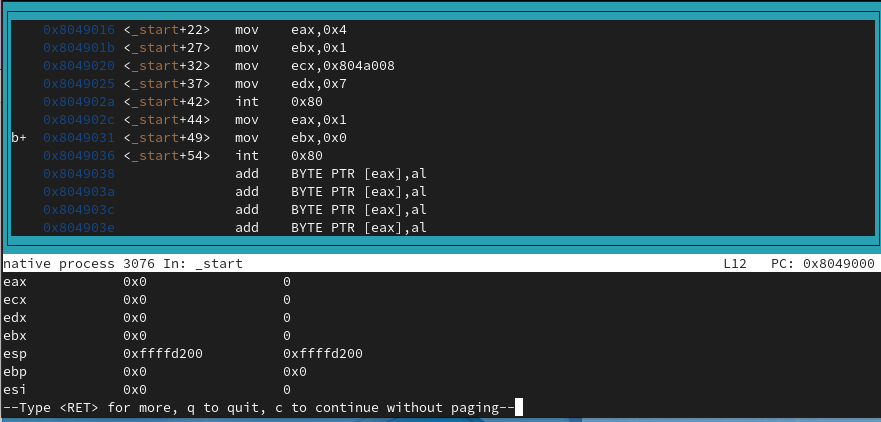


Рис. 17: Содержимое регистров

1. Посмотрела значение переменной msg1. (рис. 18)

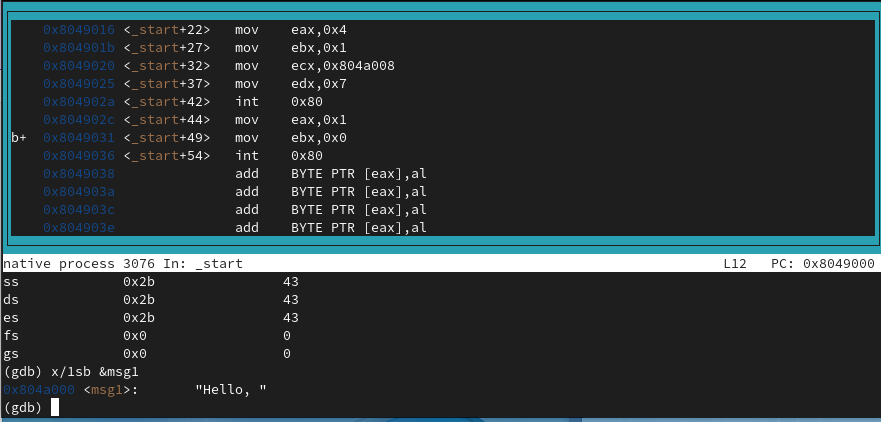


Рис. 18: Значение переменной

1. Изменила первый символ переменной msg1. (рис. 19)

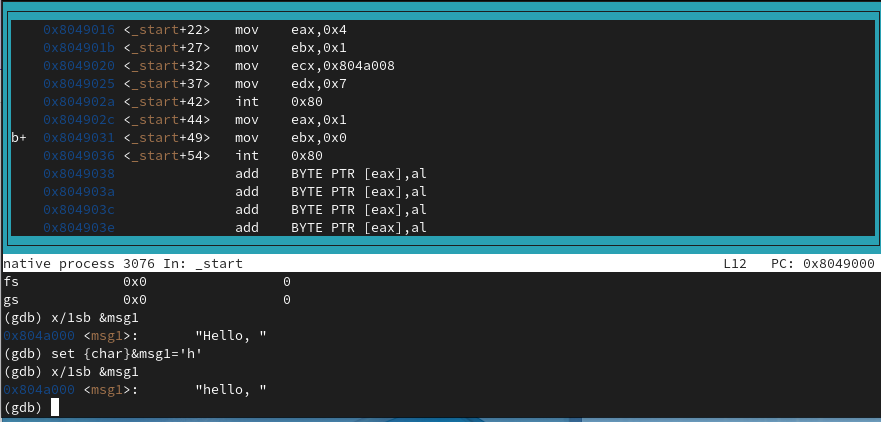


Рис. 19: Замена символа

1. Посмотрела значение переменной msg2 по адресу и изменила первый символ переменной msg1. (рис. 20)

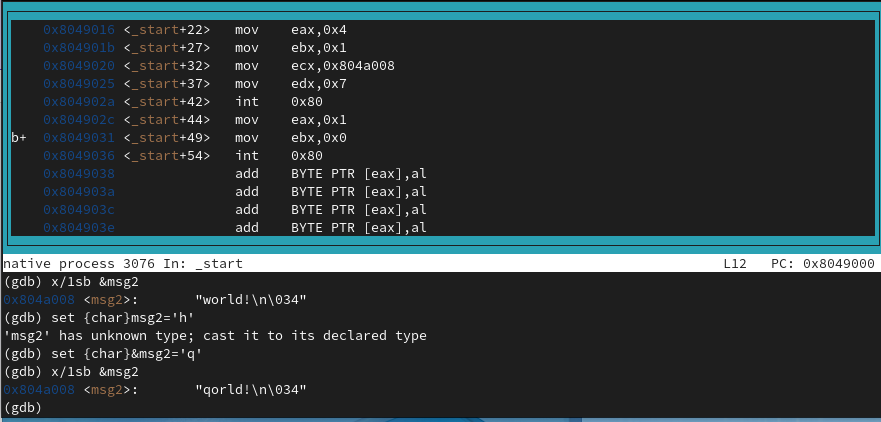


Рис. 20: Переменная msg2

1. Посмотрела значения регистров с помощью команды print /F . (рис. 21)

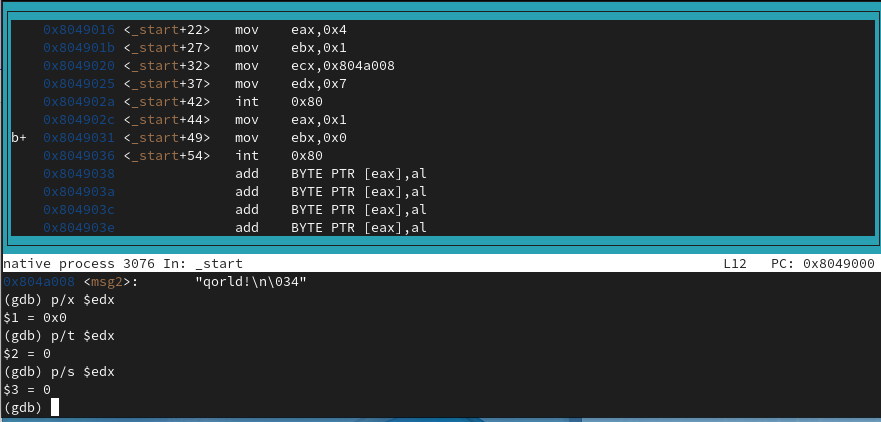


Рис. 21: Значение регистров

1. С помощью команды set изменила значение регистра ebx. (рис. 22)

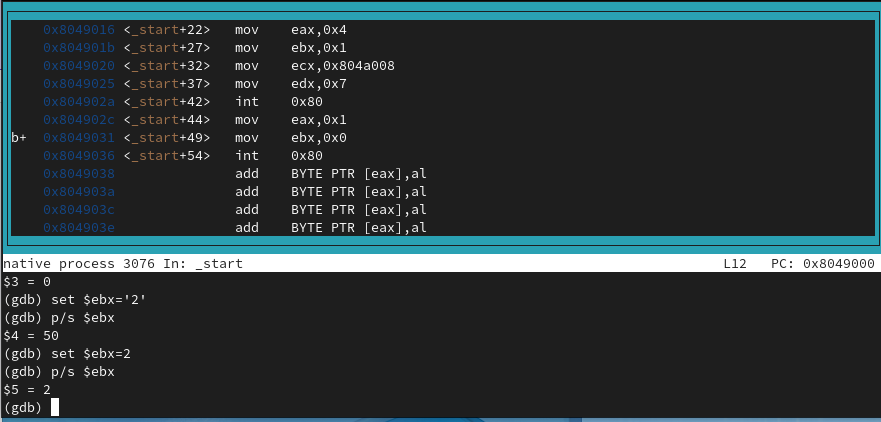


Рис. 22: Изменила значение регистра

## 4.5 Обработка аргументов командной строки в GDB

1. Скопировала файл lab9-2.asm, созданный при выполнении лабораторной работы No9, с программой выводящей на экран аргументы командной строки (Листинг 9.2) в файл с именем lab10-3.asm. (рис. 23)

Рис. 23: Копирование

Рис. 23: Копирование

1. Создала исполняемый файл. (рис. 24)

Рис. 24: Исполняемый файл

Рис. 24: Исполняемый файл

1. Использовала ключ –args. Загрузила исполняемый файл в отладчик, указав аргументы. (рис. 25)

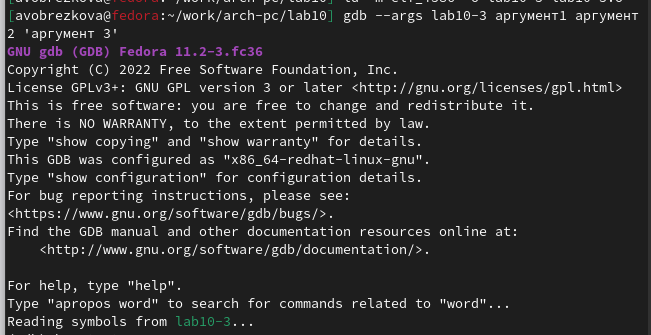


Рис. 25: Ключ –args

1. Установила точку останова перед первой инструкцией в программе и запустила ее. (рис. 26)



Рис. 26: Установка точки останова

1. Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы). (рис. 27)

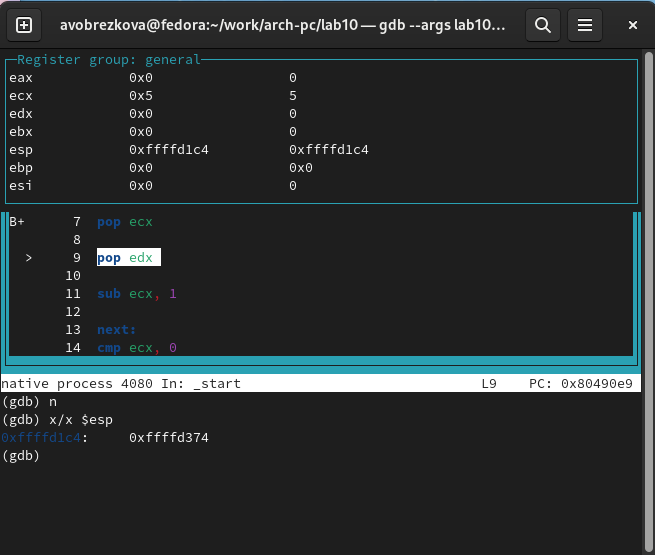


Рис. 27: .

1. Посмотрела остальные позиции стека. (рис. 28; рис. 29)

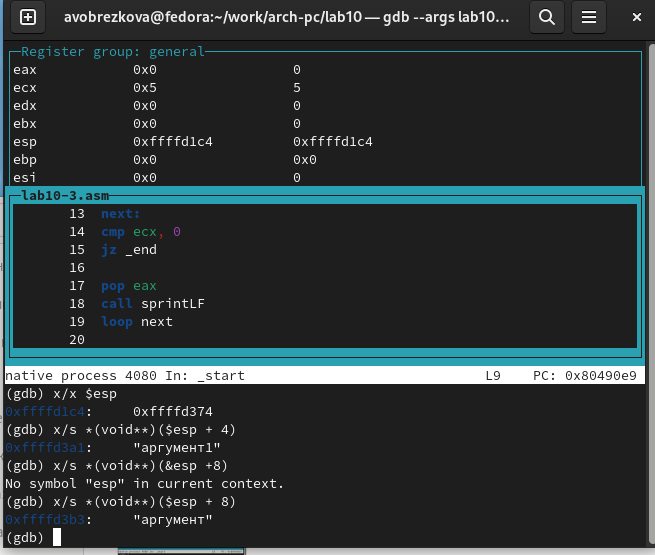


Рис. 28: Позиции стека

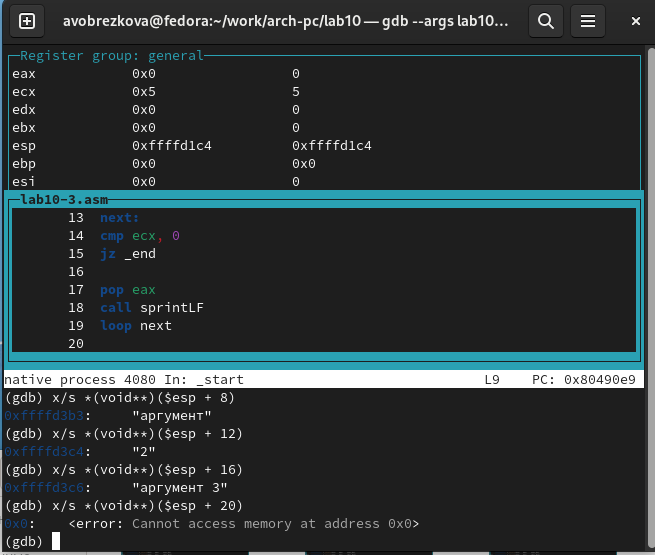


Рис. 29: Позиции стека

Шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12] и т.д.), потому что в теле цикла 4 строки кода.

## 4.6 Задания для самостоятельной работы

1. Преобразовала программу из лабораторной работы No9 (Задание No1 для самостоятельной работы), реализовав вычисление значения функции f(x) как подпрограмму. (рис. 30; рис. 31; рис. 32)



Рис. 30: Текст программы

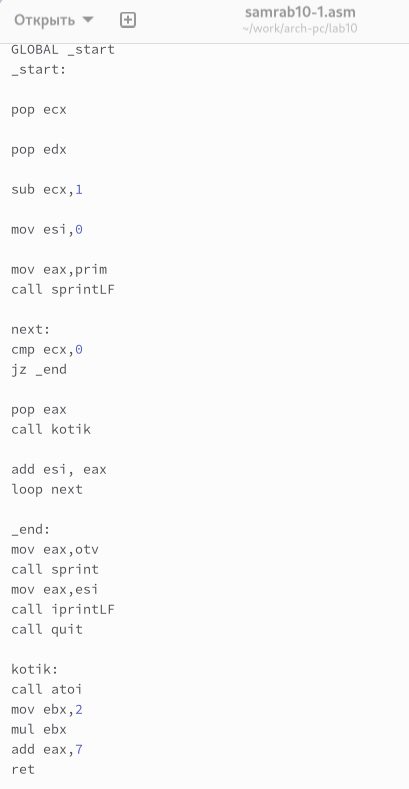


Рис. 31: Текст программы

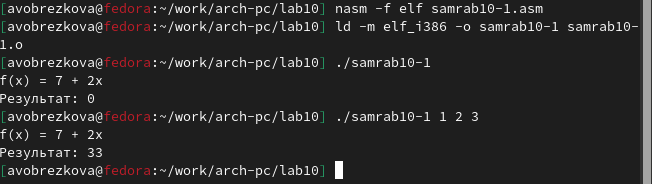


Рис. 32: Результат

1. В листинге 10.3 приведена программа вычисления выражения (3 + 2) ∗ 4 + 5. При запуске данная программа дает неверный результат. Проверила это с помощью отладчика GDB, анализируя изменения значений регистров, определила ошибку и исправила ее. (рис. 33; рис. 34; рис. 35; рис. 36)

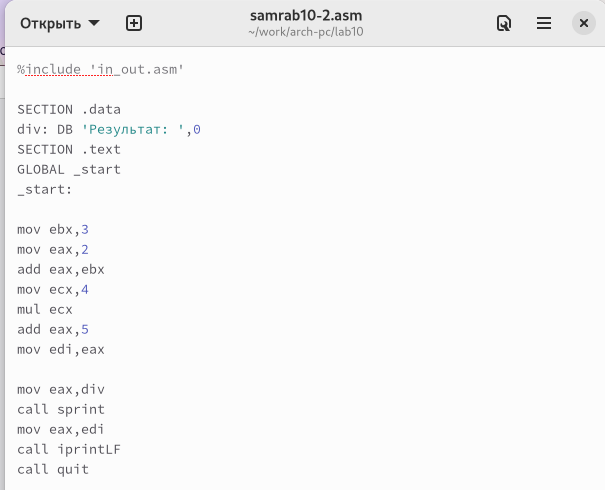


Рис. 33: Начальный текст программы

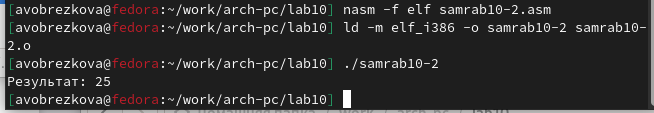


Рис. 34: Результат начальной программы

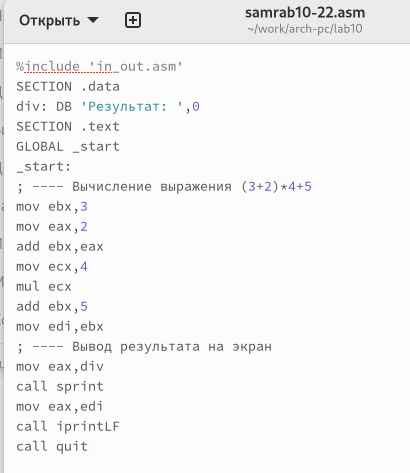


Рис. 35: Измененная программа

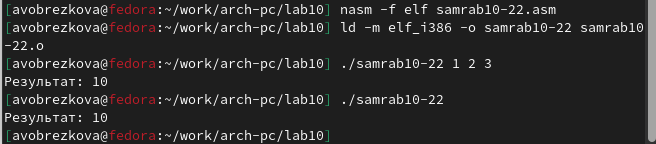


Рис. 36: Результат

Данные изменения можно проверить по ссылке: <https://github.com/avobrezkova/study_2022-2023_arh-pc/tree/master/labs/lab10>

# 5 Выводы

Приобрела навыки написания программ с использованием подпрограмм. Ознакомилась с методами отладки при помощи GDB и его основными возможностями.

# Список литературы

1. <https://esystem.rudn.ru/pluginfile.php/1584394/mod_resource/content/1/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%9610.pdf>