Лабораторная работа номер 9

Понятие подпрограммы. Отладчик GDB.

Соловьев Богдан Михайлович

Содержание

# 1 Цель работы

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

# 2 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.

Можно выделить следующие типы ошибок:

• синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; • семантические ошибки — являются логическими и приводят к тому, что программа запускается, отрабатывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль).

Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга. Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы.

Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново.

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. Пошаговое выполнение — это выполнение программы с остановкой после каждой строчки, чтобы программист мог проверить значения переменных и выполнить другие действия. Точки останова — это специально отмеченные места в программе, в которых программаотладчик приостанавливает выполнение программы и ждёт команд. Наиболее популярные виды точек останова:

• Breakpoint — точка останова (остановка происходит, когда выполнение доходит до определённой строки, адреса или процедуры, отмеченной программистом); • Watchpoint — точка просмотра (выполнение программы приостанавливается, если программа обратилась к определённой переменной: либо считала её значение, либо изменила его).

Точки останова устанавливаются в отладчике на время сеанса работы с кодом программы, т.е. они сохраняются до выхода из программы-отладчика или до смены отлаживаемой программы

# 3 Выполнение лабораторной работы

Создаю файл lab09-1.asm (рис. [1](#fig:001)).

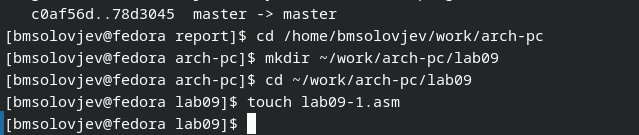


Figure 1: Создание файла lab09-1.asm

Ввожу в созданный файл код программы, считающей выражение 2𝑥 + 7 с помощью подпрограммы \_calcul.(рис. [2](#fig:002)).

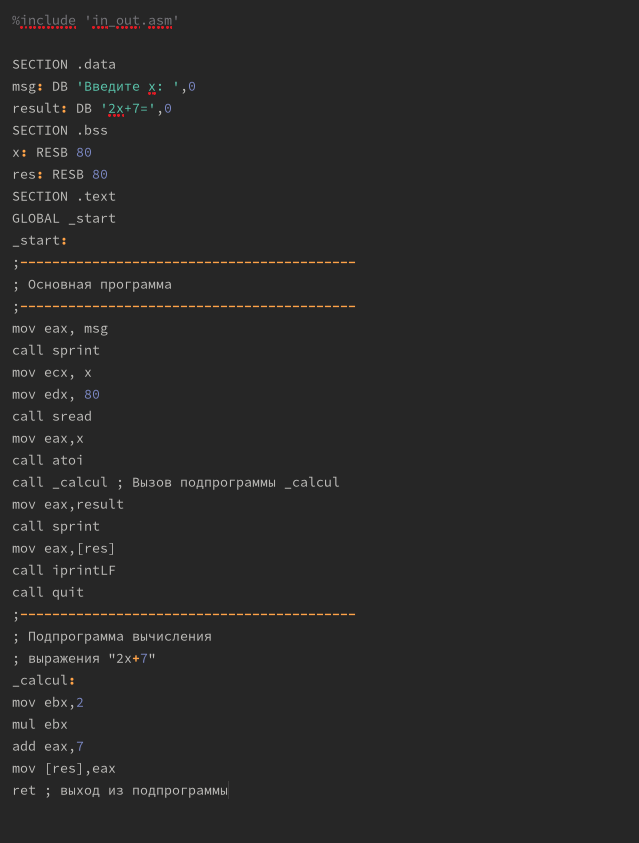


Figure 2: Код программы lab09-1.asm

Проверяю код (рис. [3](#fig:003)).

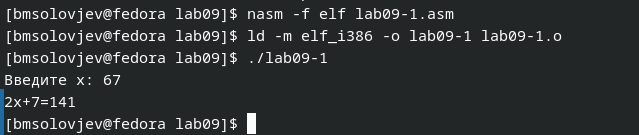


Figure 3: Работа программы lab09-1.asm

Изменяю текст программы таким образом, чтобы подпрограмма \_subcalcul считала 3𝑥 − 1 (рис. [4](#fig:004)).



Figure 4: Код программы lab09-1.asm с \_subcalcul

Проверяю код (рис. [5](#fig:005)).

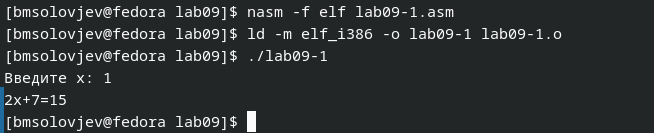


Figure 5: Работа программы lab09-1.asm с \_subcalcul

Создаю новый файл lab09-2.asm (рис. [6](#fig:006)).

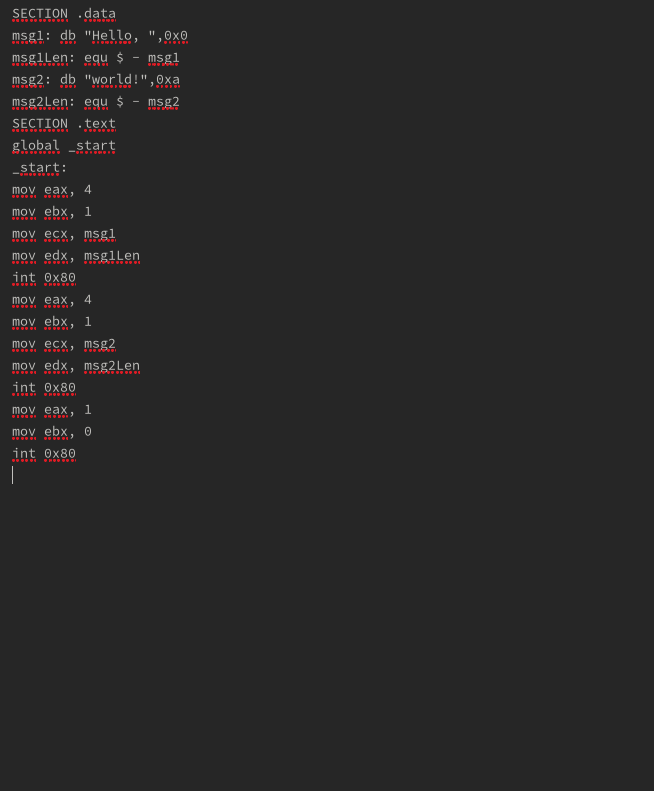


Figure 6: Код программы lab09-2.asm

Проверяю работу программы, запустив ее в оболочке GDB с помощью команды run (1) (рис. [7](#fig:007)).

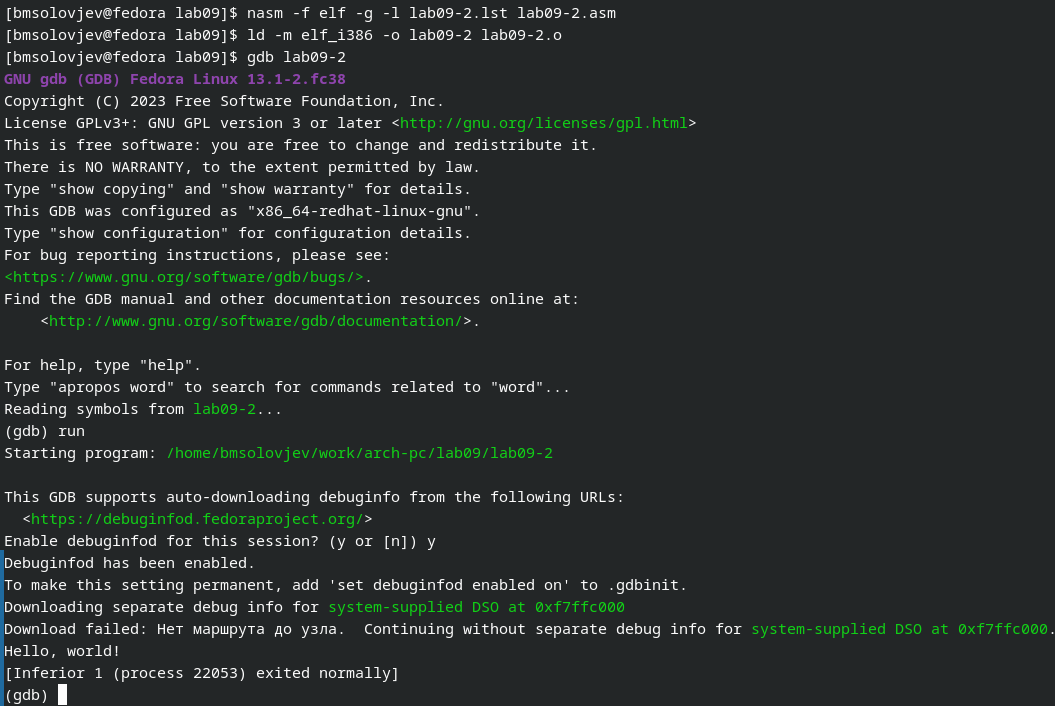


Figure 7: Проверка программы в оболочке GBD

Проверяю работу программы, запустив ее в оболочке GDB с помощью команды run (2) (рис. [8](#fig:008)).

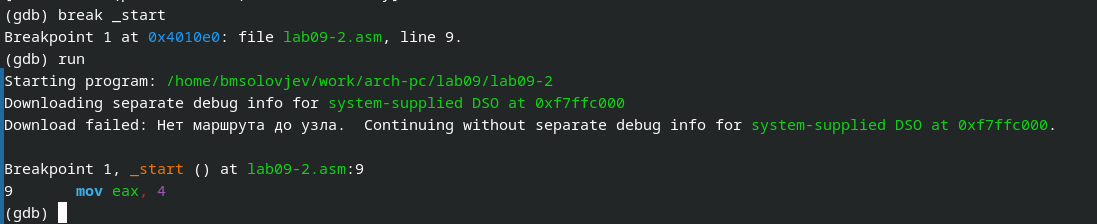


Figure 8: Код программы lab09-2.asm (2)

Сравниваю различия отображения синтаксиса машинных команд в режимах ATT и Intel. Отличие есть только в порядке отображения адреса регистра и самого регистра(рис. [9](#fig:009)).

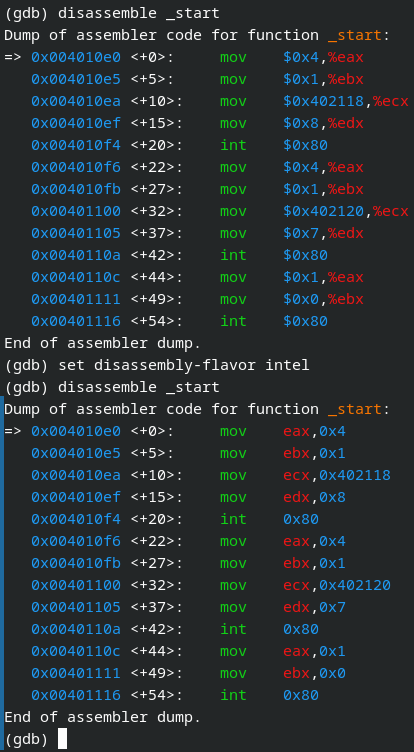


Figure 9: Отображение синтаксиса в разных режимах

Включаю режим псевдографики для более удобного анализа программы (1)(рис. [10](#fig:010)).

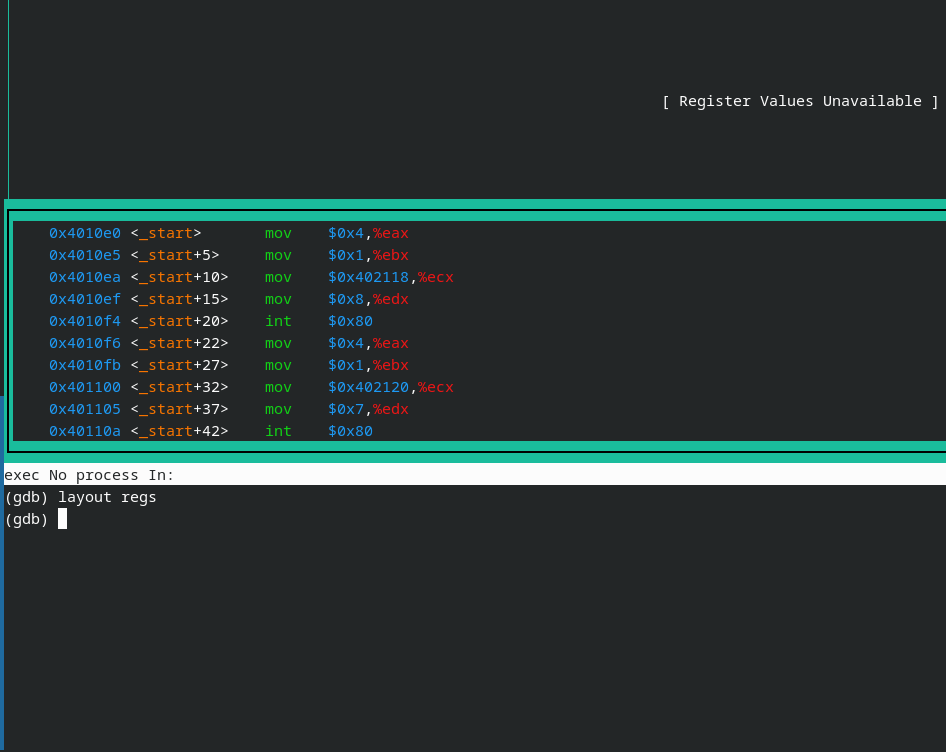


Figure 10: Режим псевдографики

Включаю режим псевдографики для более удобного анализа программы (2) (рис. [11](#fig:011)).

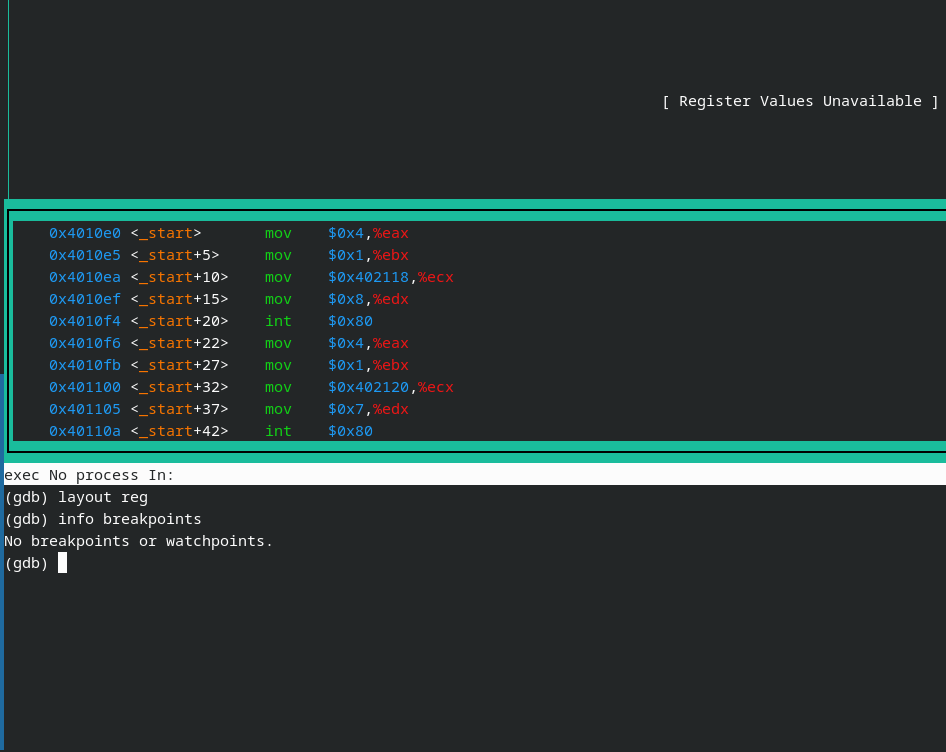


Figure 11: Режим псевдографики с регистрами

Устанавливая break point (рис. [12](#fig:012)).

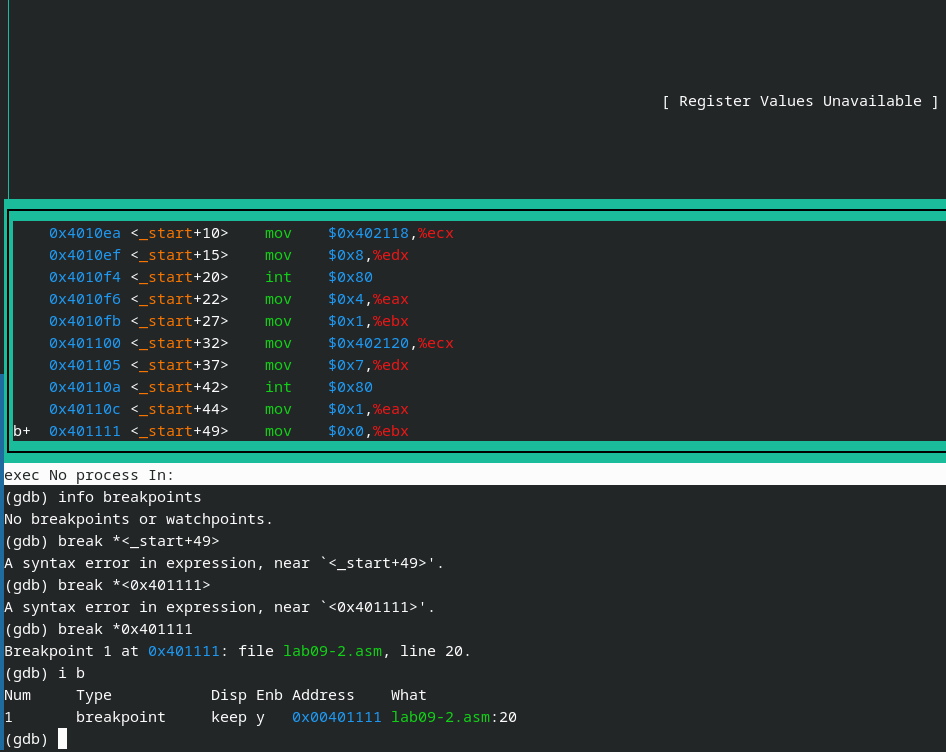


Figure 12: Установка break point

Просматриваю значение msg1 по имени (рис. [13](#fig:013)).

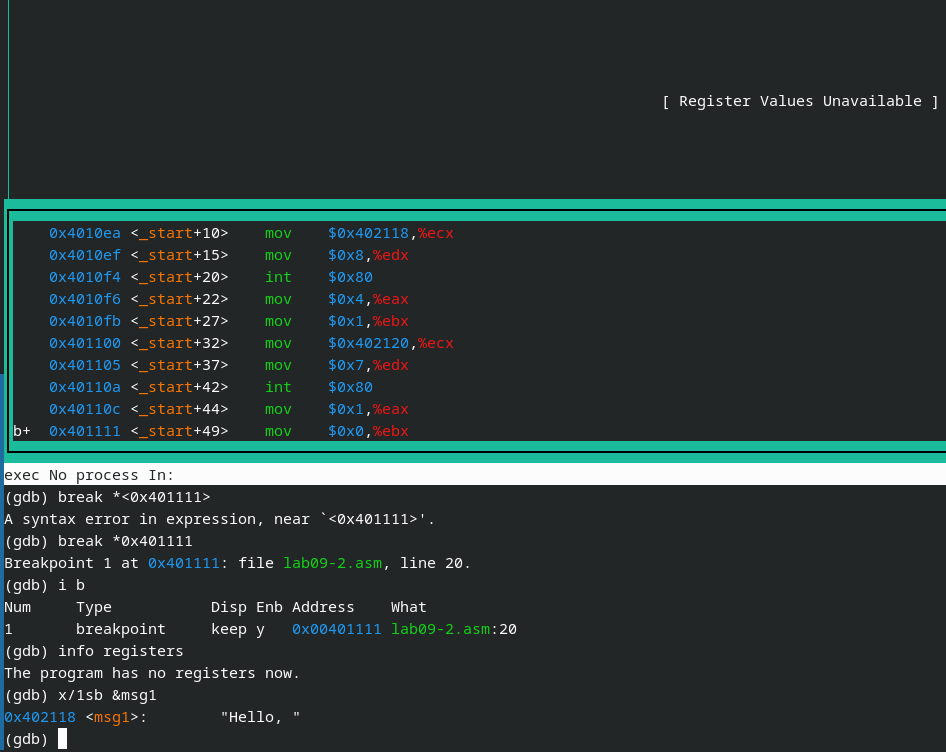


Figure 13: Просматривание значение по имени

Просматриваю значение msg2 по имени (рис. [14](#fig:014)).

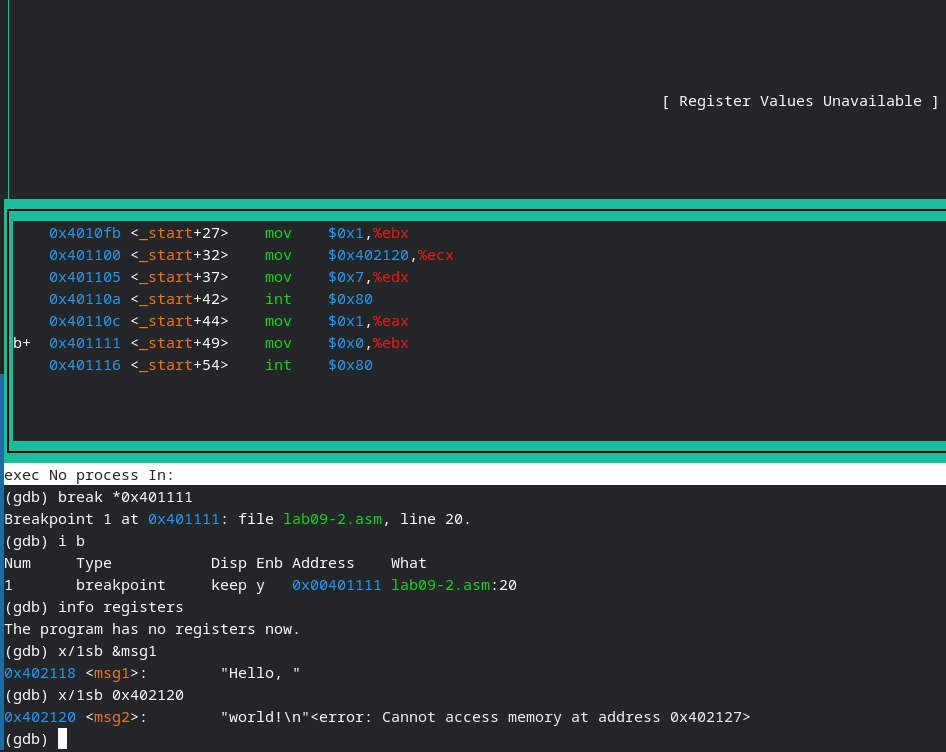


Figure 14: Просматривание значение по имени часть 2

Пытаюсь заменить символ (почему-то выдаёт ошибку )(рис. [15](#fig:015)).

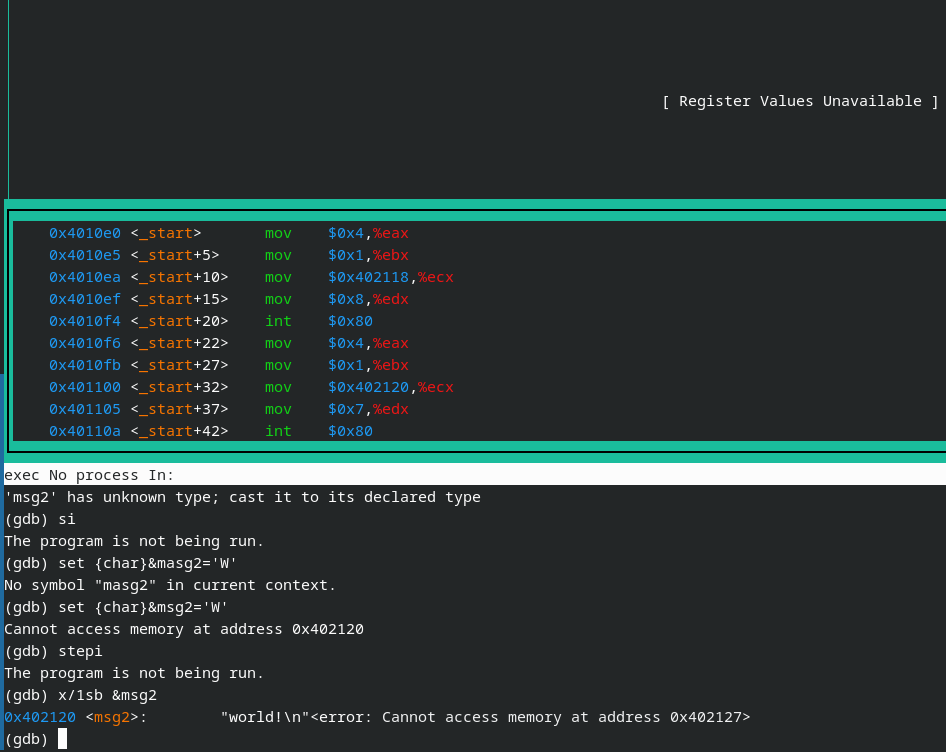


Figure 15: Ошибка

Повторяю действия из конспекта, потом загружаю в gdb программу с аргументами командой gdb –args lab09-3 аргумент1 аргумент 2 ‘аргумент 3’ (рис. [16](#fig:016)).

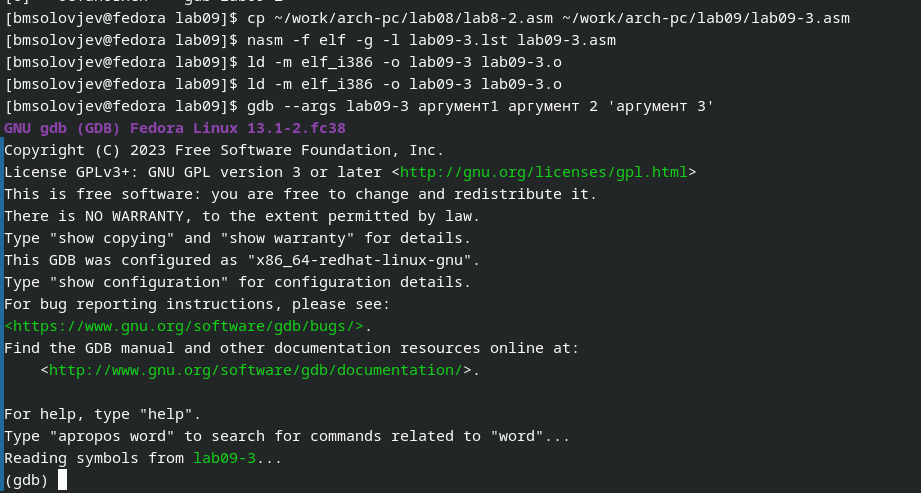


Figure 16: Программы в gbd с аргументами

Узнаю адрес вершины стека, в которого загрузились аргументы (рис. [17](#fig:017)).



Figure 17: Адрес вершины стека

Смотрю остальные позиции из стека. Их адреса находятся на одинаковом, потому что программа заранее не знает сколько места займёт каждая переменная, и поэтому места выделено на каждый элемент одинаково (рис. [18](#fig:018)).



Figure 18: Адреса в стеке

# 4 Самостоятельная работа

Я изменил код программы из прошлой лабораторной таким образом, чтобы там была подпрограмма (рис. [19](#fig:019)).

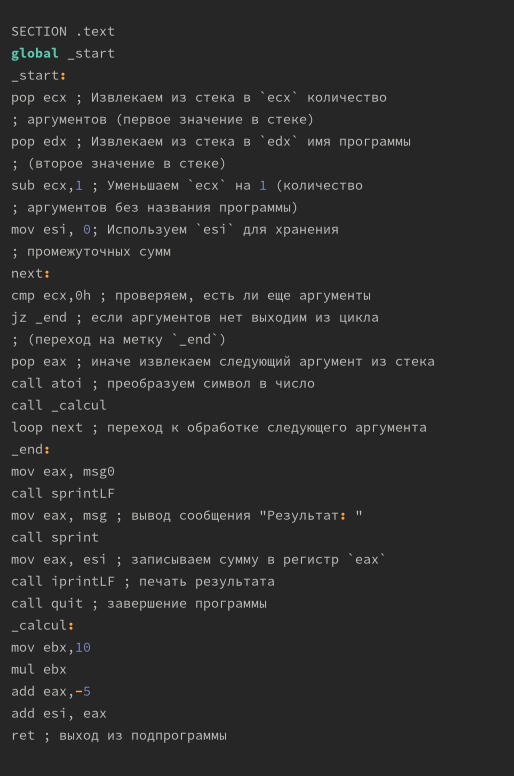


Figure 19: Код программы с подпрограммой

Проверяю правильность кода (рис. [20](#fig:020)).

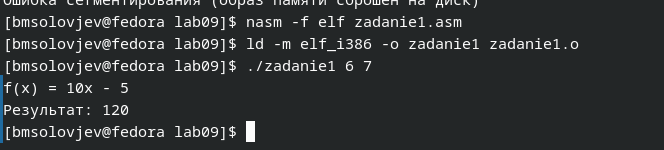


Figure 20: Проверка кода

Чтобы код работал правильно, мне пришлось заменить add ebx, eax на add eax,ebx, а mov edi, ebx на mov edi, eax (рис. [21](#fig:021)).

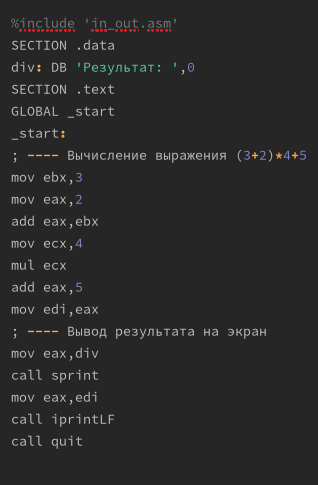


Figure 21: Измененный код

Проверка правильност кода (рис. [22](#fig:022)).

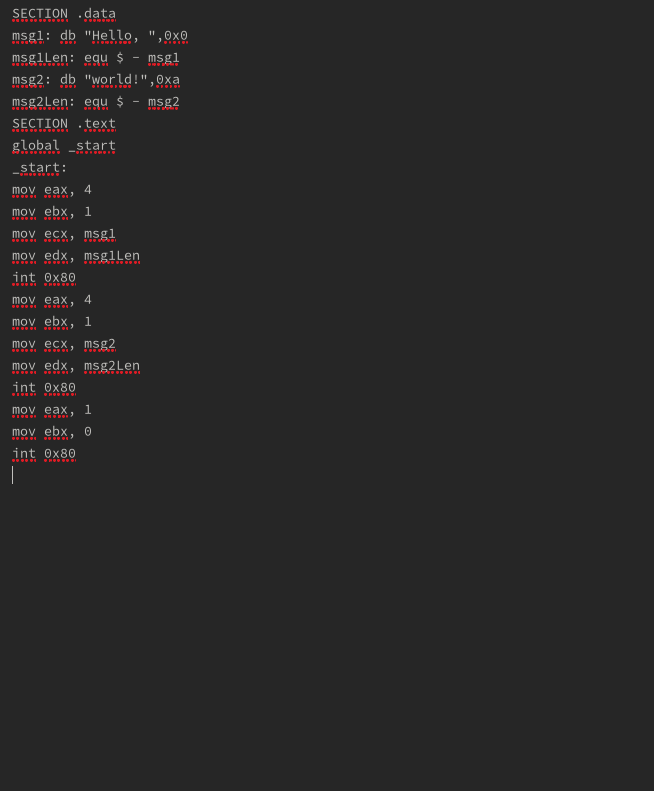


Figure 22: Код работает корректно

# 5 Выводы

Выполнив данную лабораторную работу, я приобрёл навыки написания программ с использованием подпрограмм. Познакомился с методами отладки при помощи GBD и его основными возможностями.

# Список литературы