Отчёта по лабораторной работе №9

Понятие подпрограммы. Отладчик GDB.

Ромицына Анастасия Романовна

Содержание

# 1 Цель работы

Познакомиться с методами отладки при помощи GDB, его возможностями.

# 2 Выполнение лабораторной работы

## 2.1 Реализация подпрограмм в NASM

Создаем каталог для программ Лабораторных работ, и в нем создаем файл (рис. [1](#fig:001)).

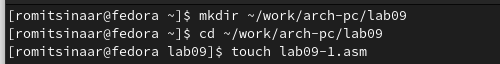


Figure 1: Создаем каталог с помощью команды mkdir и файл с помощью команды touch

Открываем файл с помощью gedit и заполняем его в соответствии с листингом 9.1 (рис. [2](#fig:002)).

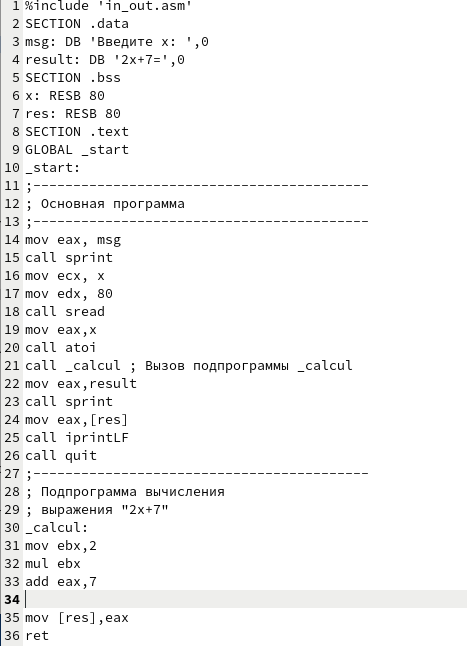


Figure 2: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. [3](#fig:003)).

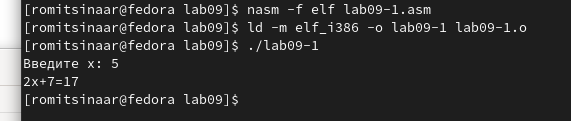


Figure 3: Запускаем файл и проверяем его работу

Открываем файл для редактирования и изменяем его, добавив подпрограмму в подпрограмму (рис. [4](#fig:004)).

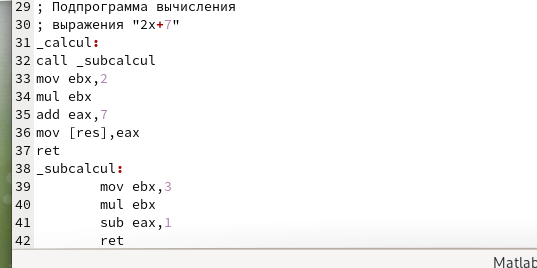


Figure 4: Изменяем файл, добавляя еще одну подпрограмму

Создаем исполняемый файл и запускаем его (рис. [5](#fig:005)).

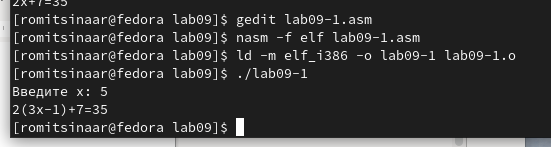


Figure 5: Запускаем файл и смотрим на его работу

## 2.2 Отладка программам с помощью GDB

Создаем новый файл в каталоге(рис. [6](#fig:006)).

Figure 6: Создаем файл

Figure 6: Создаем файл

Открываем файл и заполняем его в соответствии с листингом 9.2 (рис. [7](#fig:007)).

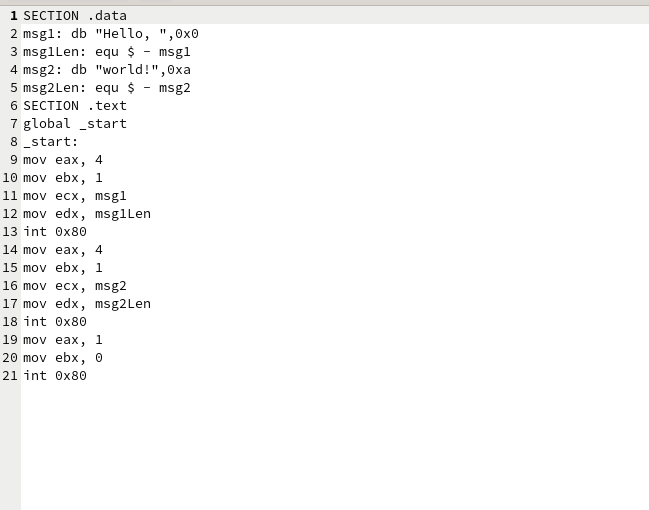


Figure 7: Заполняем файл

Получаем исходный файл с использованием отладчика gdb (рис. [8](#fig:008)).

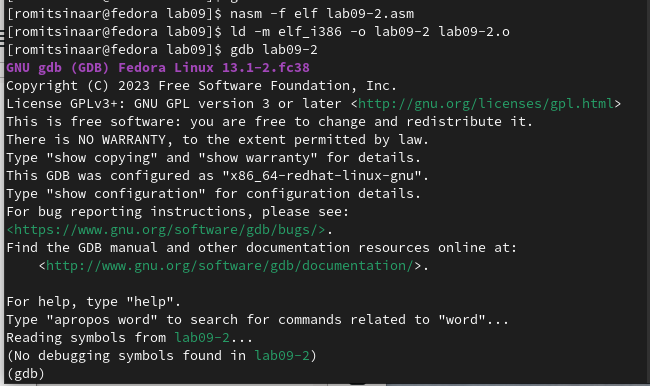


Figure 8: Загружаем исходный файл в отладчик

Запускаем команду в отладчике (рис. [9](#fig:009)).

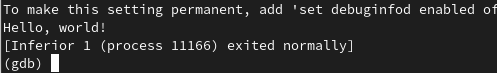


Figure 9: Запускаем программу командой run

Устанавливаем брейкпоинт на метку \_start и запускаем программу (рис. [10](#fig:010)).

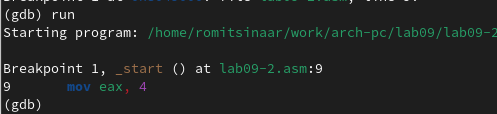


Figure 10: Запускаем программу с брейкпоином

Смотрим дисассимилированный код программы с помощью команды disassemble, начиная с метки \_start(рис. [11](#fig:011)).

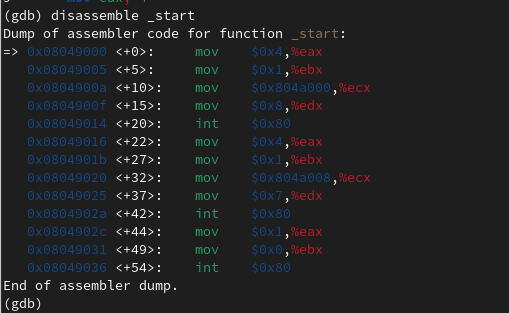


Figure 11: Смотрим дисассимилированный код программы

Переключаемся на отображение команд с Intel синтаксисом (рис. [12](#fig:012)).



Figure 12: Переключаемся на синтаксис Intel

Включаем режим псевдографики (рис. [13](#fig:013)).

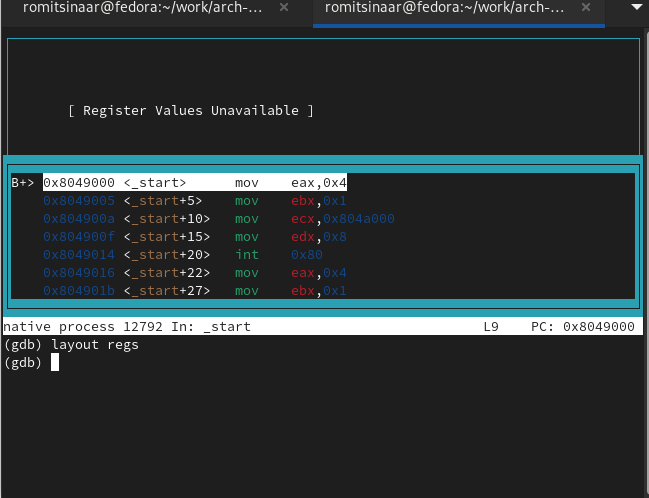


Figure 13: Включаем отображение регистров, их значений и результат дисассимилирования программы

Проверяем была ли установлена точка останова и устанавливаем точку останова предпоследней инструкции (рис. [14](#fig:014)).

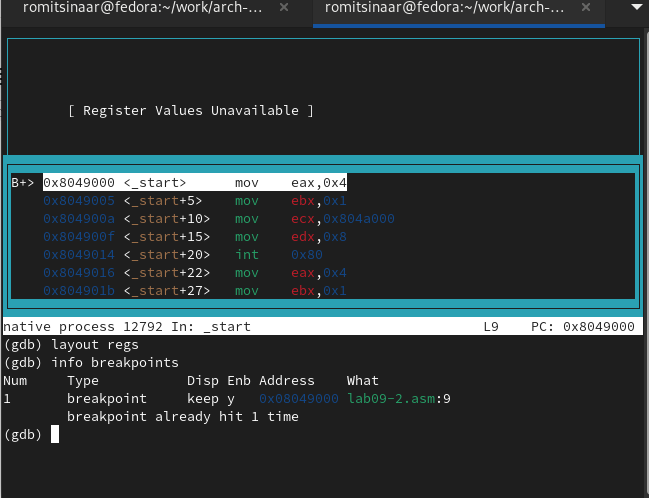


Figure 14: Используем команду info breakpoints и создаем новую точку останова

Посмотрим информацию о всех установленных точках останова (рис. [15](#fig:015)).

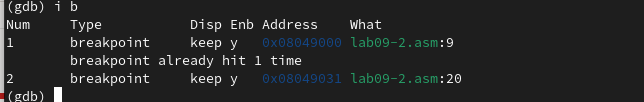


Figure 15: Смотрим информацию

Выполняем 5 инструкций командой si (рис. [16](#fig:016)).

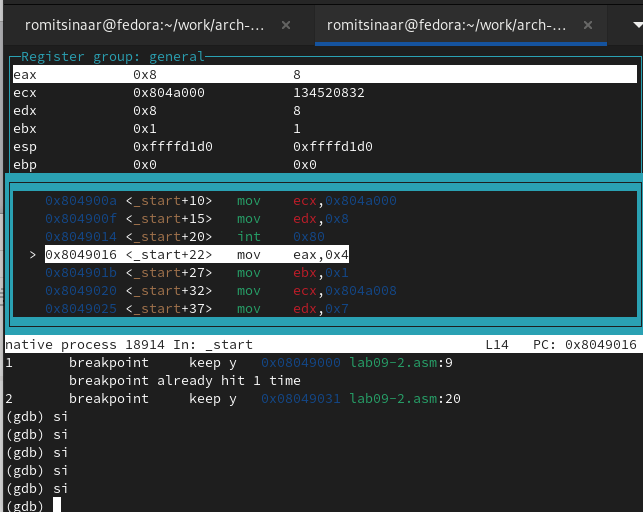


Figure 16: Отслеживаем регистры

Во время выполнения команд менялись регистры: ebx, ecx, edx,eax, eip.

Смотрим значение переменной msg1 по имени (рис. [17](#fig:017)).

Figure 17: Смотрим значение переменной

Figure 17: Смотрим значение переменной

Смотрим значение переменной msg2 по адресу (рис. [18](#fig:018)).

Figure 18: Смотрим значение переменной

Figure 18: Смотрим значение переменной

Изменим первый символ переменной msg1 (рис. [19](#fig:019)).

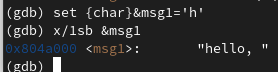


Figure 19: Меняем символ

Изменим первый символ переменной msg2 (рис. [20](#fig:020)).

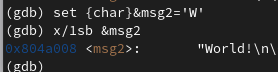


Figure 20: Меняем символ

Смотрим значение регистра edx в разных форматах (рис. [21](#fig:021)).

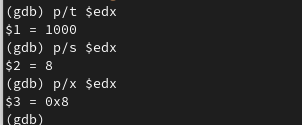


Figure 21: Смотрим значение регистра

Изменяем регистор ebx (рис. [22](#fig:022)).

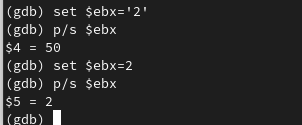


Figure 22: Изменяем регистор командой set

Выводится разные значения, так как команда без кеавычек присваивает регистру вводимое значение.

Прописываем команды для завершения программы и выхода из GDB (рис. [23](#fig:023)).



Figure 23: Прописываем команды c и quit

Копируем файл lab8-2.asm в файл с именем lab09-3.asm (рис. [24](#fig:024)).

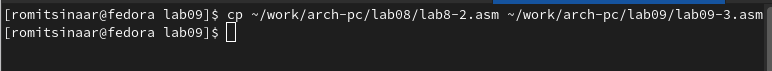


Figure 24: Копируем файл

Создаем исполняемый файл и запускаем его в отладчике GDB (рис. [25](#fig:025)).

Figure 25: Создаем и запускаем в отладчике файл

Figure 25: Создаем и запускаем в отладчике файл

Установим точку останова перед первой инструкцией в программе и запустим ее (рис. [26](#fig:026)).

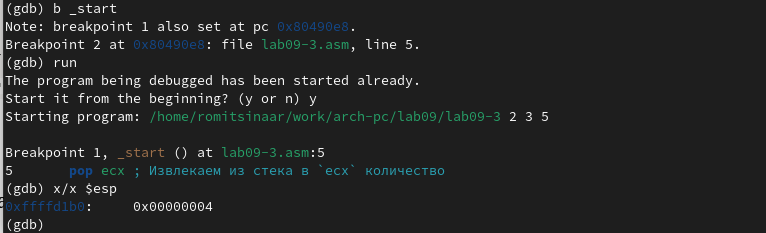


Figure 26: Устанавливаем точку останова

Смотрим позиции стека по разным адресам (рис. [27](#fig:027)).



Figure 27: Изучаем полученные данные

Шаг изменения адреса равен 4 потому что адресные регистры имеют размерность 32 бита(4 байта).

##Задание для самостоятельной работы

###Задание 1

Копируем файл lab8-4.asm(ср №1 в лабораторной работый 8) в файл с именем lab09-3.asm (рис. [28](#fig:028)).

Figure 28: Копируем файл

Figure 28: Копируем файл

Открываем файл с gedit и меняем его, создавая подпрограмму (рис. [29](#fig:029)).

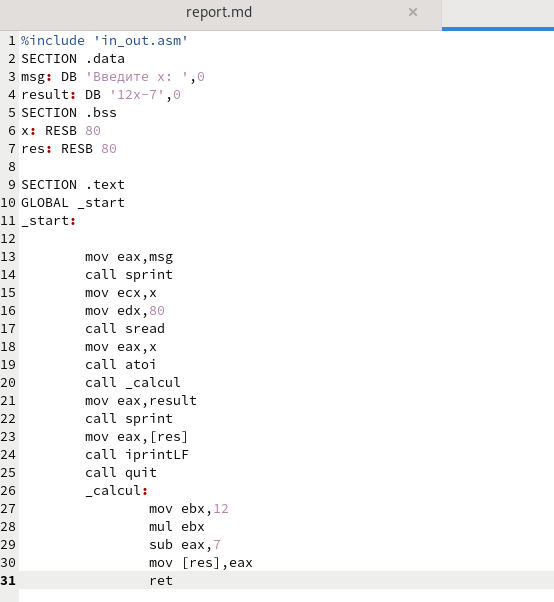


Figure 29: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. [30](#fig:030)).

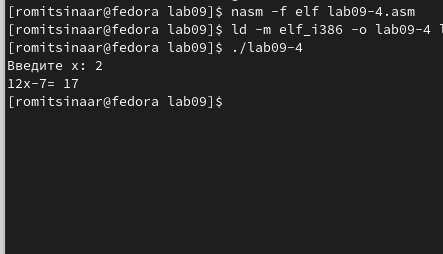


Figure 30: Проверяем работу программы

###Задание 2

Создаем новый файл в дирректории (рис. [31](#fig:031)).

Figure 31: Создаем файл

Figure 31: Создаем файл

Открываем файл с помощью gedit и заполняем его в соответствии с листингом 9.3 (рис. [32](#fig:032)).

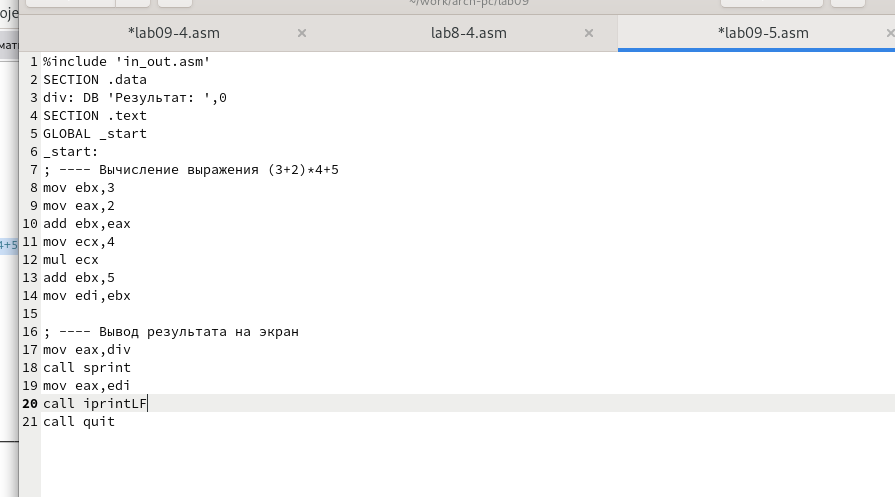


Figure 32: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. [33](#fig:033)).

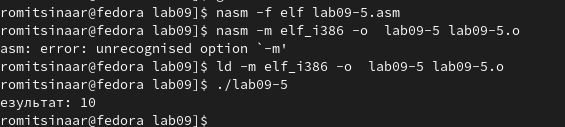


Figure 33: Создаем и смотрим на работу программы(работает неправильно)

Создаем исполняемый файл и запускаем его в отладчике gdb и смотрим на изменение решистров командой si (рис. [34](#fig:034)).

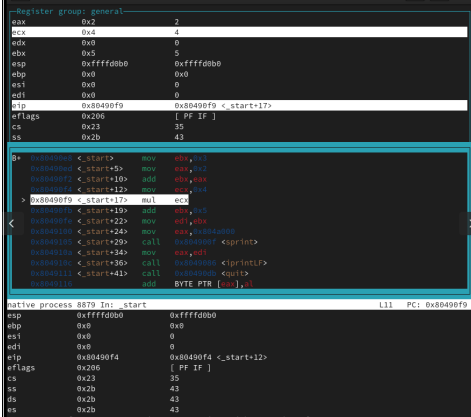


Figure 34: Ищем ошибку регистров в отладчике

Изменяем программу для корректной работы (рис. [35](#fig:035)).

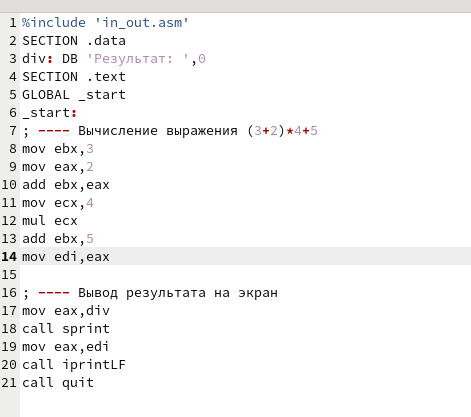


Figure 35: Меняем файл

Создаем исполняемый файл и запускаем его (рис. [36](#fig:036)).

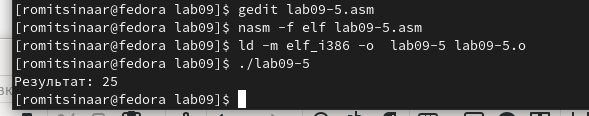


Figure 36: Создаем и запускаем файл(работает корректно)

# 3 Выводы

Мы познакомились с методами отладки при помощи GDB и его возможностями.