Отчет по лабораторной работе №8

Дисциплина: Архитектура компьютера

Стрижов Дмитрий Павлович

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. Реализация подпрограмм в NASM
2. Отладка программам с помощью GDB
3. Задание для самостоятельной работы

# 3 Выполнение лабораторной работы

## 3.1 Реализация подпрограмм в NASM

Создаю каталог для выполнения лабораторной работы № 9, перехожу в него и создаю файл lab09-1.asm (рис. [1](#fig:001)).

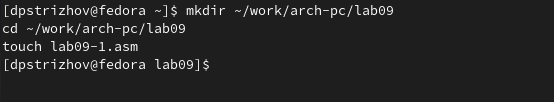


Figure 1: Предварительная подготовка

Копирую программу из листинга 1, добавляю подпрограмму \_subcalcul, которая вычисляет функцию g(x) (рис. [2](#fig:002)).



Figure 2: Реализация подпрограммы \_subcalcul

Вот что получается в итоге (рис. [3](#fig:003)).

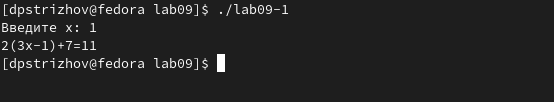


Figure 3: Вывод программы с подпрограммой \_subcalcul

## 3.2 Отладка программам с помощью GDB

Создаю файл lab09-2.asm, куда вписываю программу из листинга 2 (рис. [4](#fig:004), рис. [5](#fig:005)).

Figure 4: Создание lab09-2.asm

Figure 4: Создание lab09-2.asm

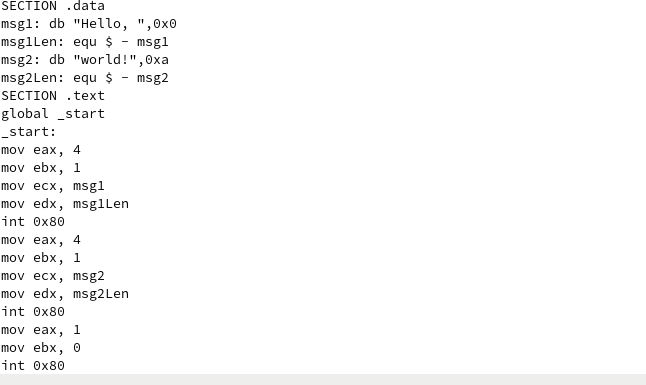


Figure 5: Содержимое lab09-2.asm

Получаю исполняемый файл, при этом указывая ключ “-g” для работы с GDB (рис. [6](#fig:006)).

Figure 6: Подготовка для работы с отладчиком

Figure 6: Подготовка для работы с отладчиком

Загружаю исполняемый файл в отладчик (рис. [7](#fig:007)).

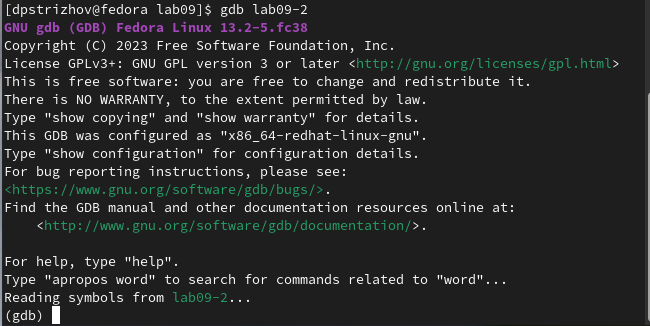


Figure 7: Загрузка исполняемого файла в отладчик

Проверяю работы программы (рис. [8](#fig:008)).

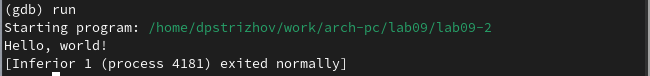


Figure 8: Проверка раоты программы

Ставим метку и снова запускаем программу (рис. [9](#fig:009)).

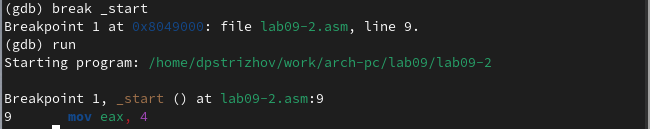


Figure 9: Проверка программу с меткой

Просматриваю дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. [10](#fig:010)).

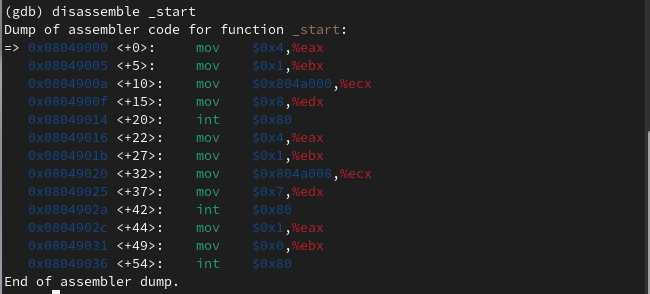


Figure 10: Дисассимилированный код программы

Переключаюсь на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel, различия отображения в синтаксисе ATT, а именно знак % перед регистром и регистры при написании программ меняются местами, например, в при выполнении команды mov значения из первого регистра переходят во второй (рис. [11](#fig:011)).



Figure 11: Переключание на отображение команд с Intel’овским синтаксисом

Включаю режим псевдографики для более удобного анализа программы (рис. [12](#fig:012)).

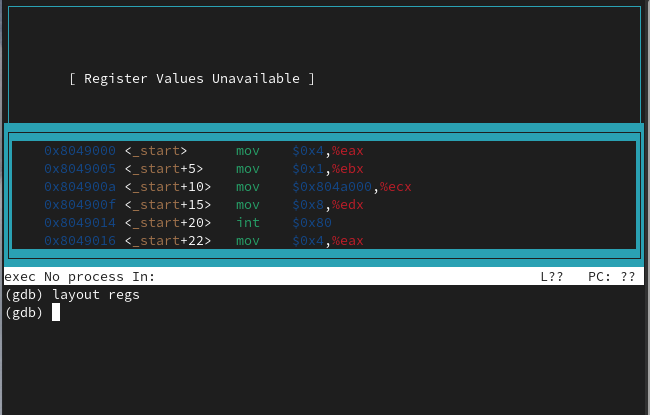


Figure 12: Режим псевдографики

Проверяю точку останова (рис. [13](#fig:013)).



Figure 13: Проверка точки останова

Установка новой точки останова (рис. [14](#fig:014)).

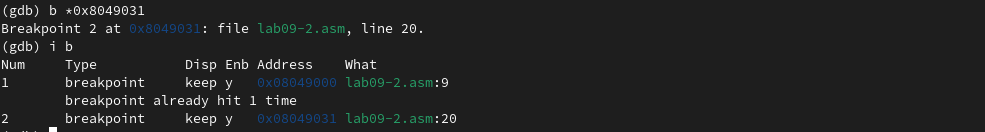


Figure 14: Новая точка останова

Смотрю содержимое переменной msg1 (рис. [15](#fig:015)).

Figure 15: Содержимое переменной msg1

Figure 15: Содержимое переменной msg1

Смотрю значение переменной msg2 (рис. [16](#fig:016)).

Figure 16: Значение переменной msg2

Figure 16: Значение переменной msg2

Изменяем значения переменной msg1 (рис. [17](#fig:017)).

Figure 17: Измененнная переменная msg1

Figure 17: Измененнная переменная msg1

Делаю то же самое с переменной msg2 (рис. [18](#fig:018)).

Figure 18: Измененная переменная msg2

Figure 18: Измененная переменная msg2

Меняю значение регистра ebx (рис. [19](#fig:019)).

Figure 19: Регистр ebx

Figure 19: Регистр ebx

И снова, причина разницы значений заключается в том, что в первом случае мы передаем строку, а во втором число (рис. [20](#fig:020)).

Figure 20: Изменения регистра ebx

Figure 20: Изменения регистра ebx

Копирую файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm(рис. [21](#fig:021)).

Figure 21: Подготовка к дальнейшей работе

Figure 21: Подготовка к дальнейшей работе

Загружаю программу в отладчик, устанавлюваю точку останова и запускаю программу (рис. [22](#fig:022)).

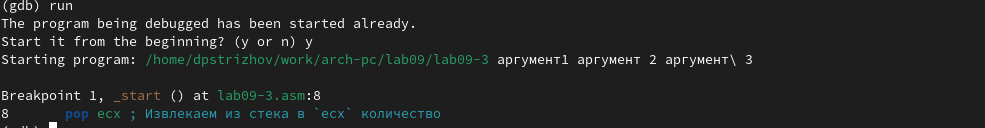


Figure 22: Работа переданной программы в отладчике

Смотрим, что находиться в регистре esp (стек) (рис. [23](#fig:023)).

Figure 23: Регистр esp

Figure 23: Регистр esp

Рассматриваем прочие значения стека (рис. [24](#fig:024)).

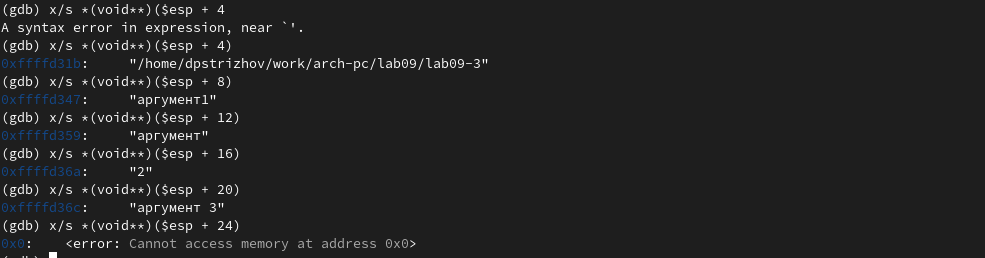


Figure 24: Прочие значения стека

Знак изменения равен 4, вероятно, потому, что адрес записывается в 16-тиричной системе исчесления.

## 3.3 Задание для самостоятельной работы

1. Меняю программу, добавляя подпрограмму, которая считает функцию отдельно (рис. [25](#fig:025)).

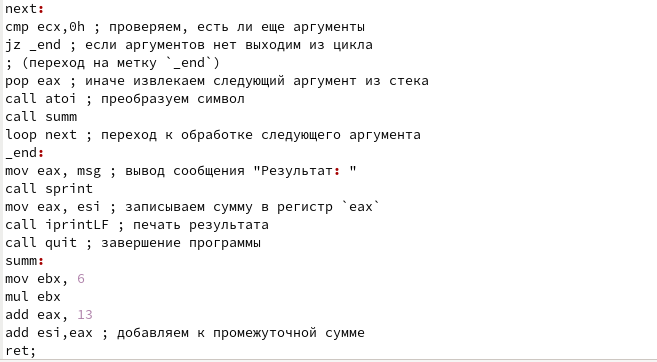


Figure 25: Подпрограмма summ

Получаю следующий результат (рис. [26](#fig:026)).

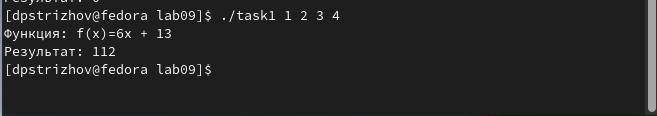


Figure 26: Результат вычислений

1. Меняю программу из листинга, чтобы выводился правильный результат (рис. [27](#fig:027)).

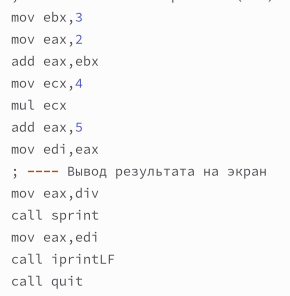


Figure 27: Испрпавленная программа

Проверяю её исполнение и получаю правильный результат (рис. [28](#fig:028)).

Figure 28: Результат исправленной программы

Figure 28: Результат исправленной программы

# 4 Выводы

За время выпонения работы я получил навыки написания программ с использованием подпрограмм и познакомился с методами отладки с помощью GDB и его основными осоьенностями.

# Список литературы

Debugging assembly with GDB. Источник: https://ncona.com/2019/12/debugging-assembly-with-gdb/