Отчёт по лабораторной работе №9

Простейший вариант

Янушкевич Михаил денисович

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. Создать каталог для ЛБ9, в нём создать файл lab09-1.asm.(рис. [1](#fig:001))

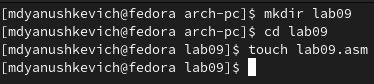


Figure 1: Создание файла

С помощью команды touch создаём файл lab09-1.asm.

1. В файл lab09-1.asm ввести программу из листинга 9.1.(рис.([2](#fig:002))

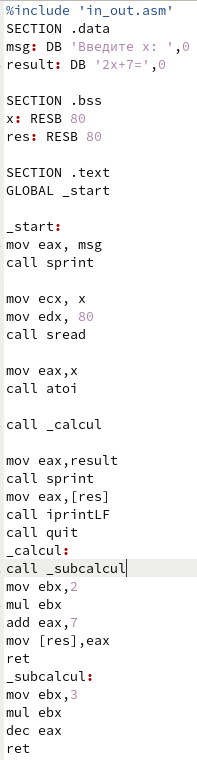


Figure 2: Ввод программы

Вводим в файл lab09-1.asm текст программы из листинга 9.1.

1. Создать исполняемый файл, проверить его работу.(рис. [3](#fig:003))

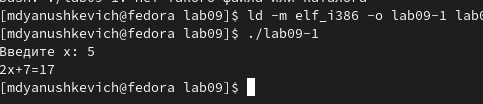


Figure 3: Создание исполняемого файла, проверка программы

Вводим необходимые команды, чтобы создать исполняемый файл. Проверяем его работу.

1. Изменить текст программы(рис. [4](#fig:004))

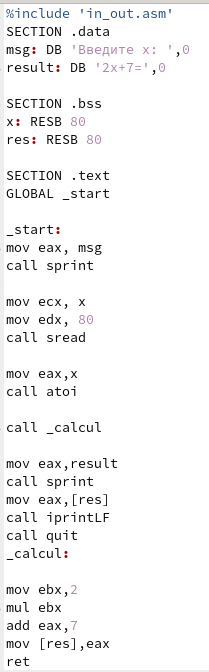


Figure 4: Изменение текста программы

в файл lab09-1.asm вносим необходимые изменения, добавив подпрограмму \_subcalcul в подпрограмму \_calcul.

1. Создать исполняемый файл, проверить его работу.(рис. [5](#fig:005))

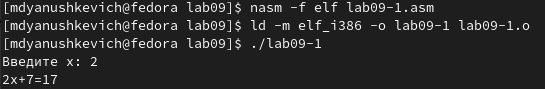


Figure 5: Создание исполняемого файла, проверка программы

Вводим необходимые команды, чтобы создать исполняемый файл. Проверяем его работу. Число 17 является правильным ответом.

1. Создать файл lab09-2.asm, ввести в него текст программы листинга 9.2.(рис. [6](#fig:006))

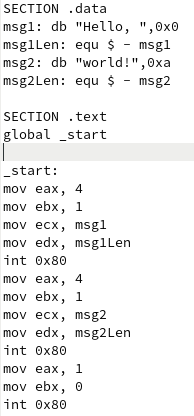


Figure 6: Ввод программы

Создаём файл lab09-2.asm, в него вводим код из листинга 9.2.

1. Получить исполняемый файл, проверить его работу.(рис. [7](#fig:007))

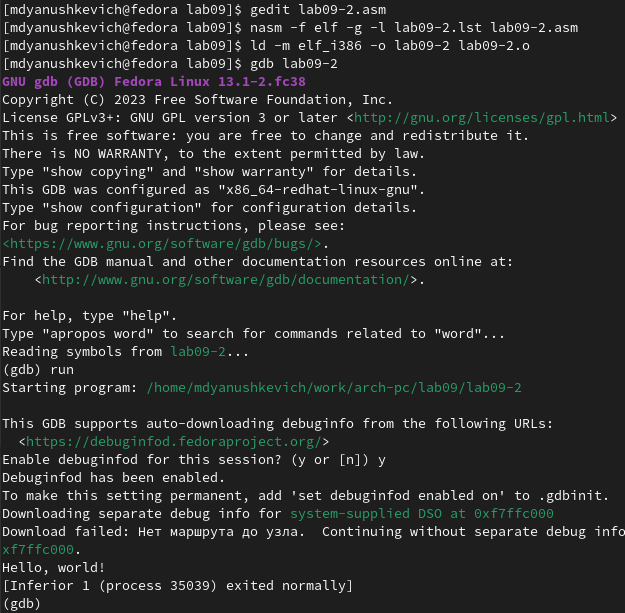


Figure 7: Создание исполняемого файла, проверка программы

Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого вводим ключ -g. Загружаем файл в отладчик GDB, запускаем с помощью команды run.

1. Установить брейкпоинт, запустить программу.(рис. [8](#fig:008))

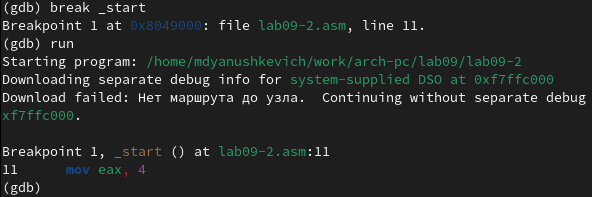


Figure 8: Установка брейкпоинта.

С помощью команды break \_start устанавливаем брейкпоинт. С помощью команды run запускаем программу.

1. Посмотреть дисассимилированный код программы, переключиться на отображение команд с INTEL’овским синтаксисом.(рис. [9](#fig:009))

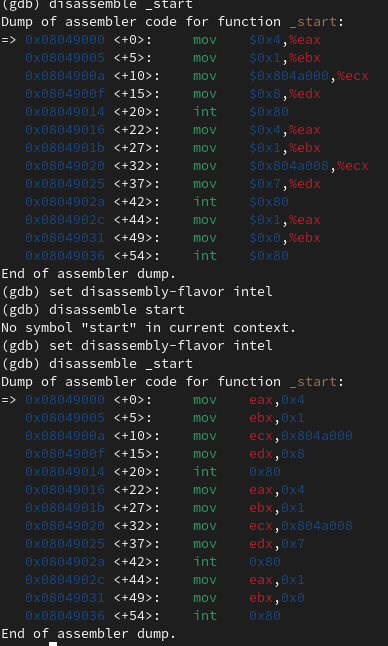


Figure 9: Просмотр дисассимилированного кода, переключение синтаксиса

С помощью команды disassemble просматриваем дисассимилированный код. Далее переключаем синтаксис с помощью команды set.

1. Включить режим псевдографики.(рис. [10](#fig:010))

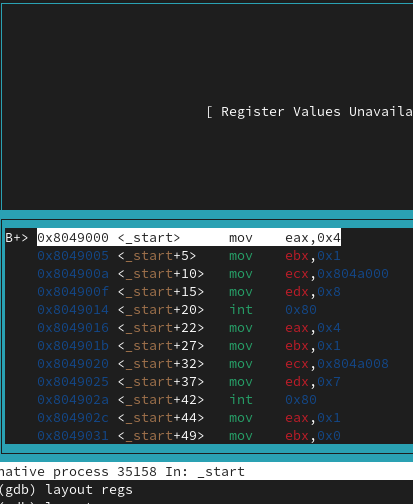


Figure 10: Включение режима псевдографики

Вводим необходимые команды для включения режима псевдографики.

1. Проверить точку останова.(рис. [11](#fig:011))

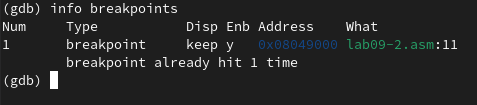


Figure 11: Проверка точки останова

С помощью команды breakpoints проверяем точку останова.

1. Определить адрес предпоследней инструкции и установаить точку останова, посмотреть информацию о всех установленных точках останова.(рис. [12](#fig:012))

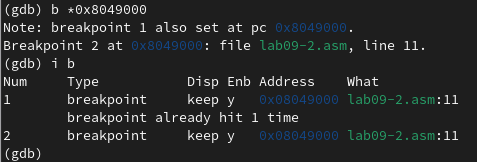


Figure 12: Определение адреса, просмотр информации

С помощью команды break устанавливаем точку останова, с помощью команды i b просматриваем информацию о других точках.

1. Выполнить 5 инструкций, проследить за изменением значений регистров.(рис. [13](#fig:013))

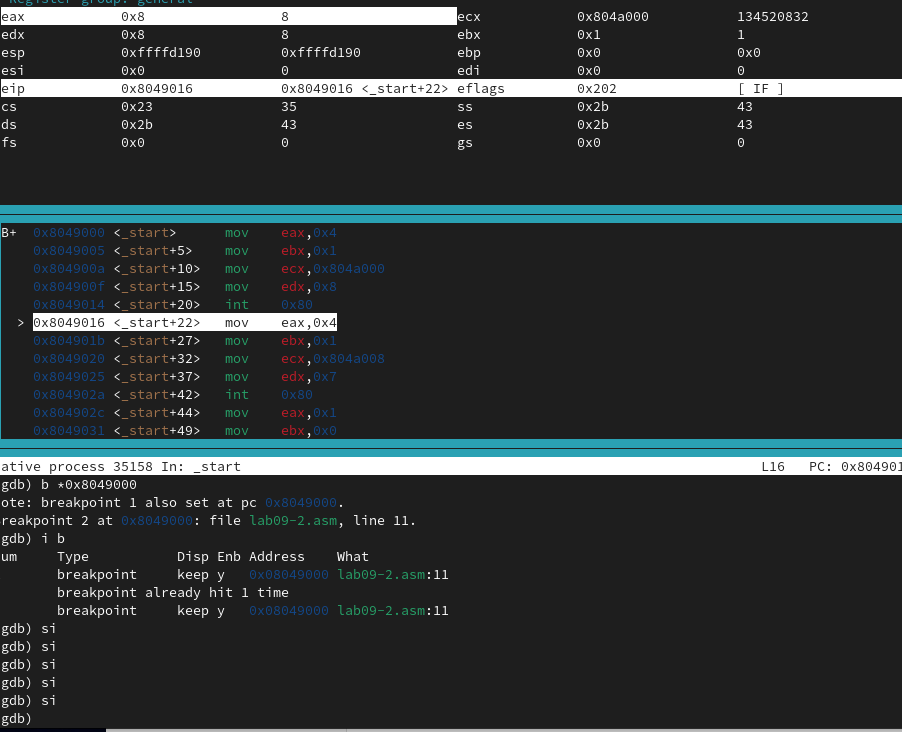


Figure 13: Выполнение инструкций

С помощью команды si просматриваем изменение значений регистров. Изменяются eax,ebx,ecx,edx.

1. Просмотреть значение переменной msg1.(рис. [14](#fig:014))

Figure 14: Просмотр значения

Figure 14: Просмотр значения

Вводим необходимую команду,смотрим значение переменной.

1. Просмотреть значение переменной msg2.(рис. [16](#fig:015))

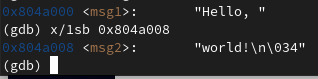


Figure 15: Просмотр значения

Вводим необходимую команду, смотрим значение переменной.

1. Изменение первого символа переменной msg1.(рис. ??)

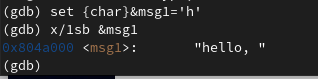


Figure 16: Изменение первого символа

С помощью команды set изменяем символ ‘H’ на ‘h’.

1. Изменение любого символа переменной msg2.(рис. [17](#fig:017))

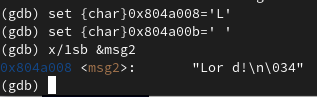


Figure 17: Изменение символа

С помощью команды set изменяем несколько символов в переменной msg2.

1. Вывести регистр edx в различных форматах.(рис. [18](#fig:018))

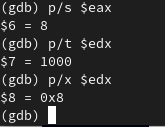


Figure 18: Вывод регистра

Выводим значение регистра edx с шестнадцатеричном, двоичном и символьном форматах.

1. Изменение значения регистра ebx.(рис. [19](#fig:019))

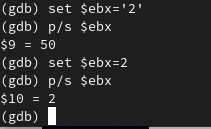


Figure 19: Изменение значения регистра

С помощью команды set изменяем значение регистра ebx.

1. Скопировать файл lab8-2.asm, создать исполняемый файл, загрузить в GDB.(рис. [20](#fig:020))

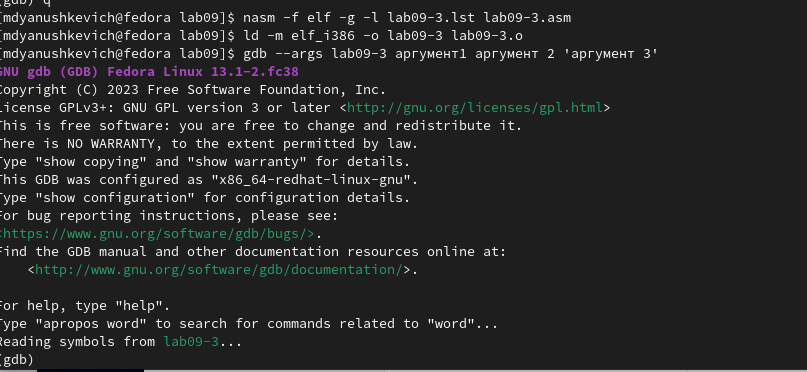


Figure 20: Копирование файла, загрузка в GDB

Копируем файл lab8-2.asm, создаём исполняемый файл, загружаем в GDB, используя ключ –args.

1. Установка точки останова, запуск инструкции.(рис. [21](#fig:021))

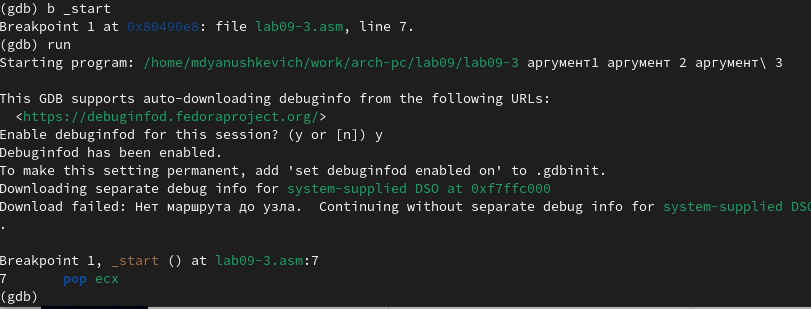


Figure 21: Установка точки

Устанавливаем точку останова, запускаем инструкцию с помощью команды run.

1. Посмотреть все позиции стека в регистре esp.(рис. [22](#fig:022))

Figure 22: Просмотр позиций стека

С помощью определенных команд просматриваем все позиции стека в регистре esp, находящиеся в памяти.

# 3 Задание для самостоятельной работы

1. Создать файл lab09-4.asm.(рис. [23](#fig:023))

Figure 23: Создание файла

С помощью команды touch создать файл lab09-4.asm.

1. Вводим в файл текст измененной программы(рис. [24](#fig:024))

Figure 24: Ввод программы

# 4 Выводы

Благодаря этой лабораторной работе я приобрёл навыки по использованию GDB, и с помощью неё смог ознакомился с методами отладки программ. Также смог написать программы с использованием подпрограмм.

# Список литературы