Отчёт по лабораторной работе

Лабораторная №10

Дикач Анна Олеговна

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

## 2.1 Реализация подпрограмм в NASM

1. создаю каталог для выполнения лабораторной работы №10, перехожу в него и создаю файл lab10-1.asm
   1. внимательно изучаю текст программы, создаю исполняемый файл и проверяю его работу. так как файл не работал, вношу некоторые правки. (рис. 1)

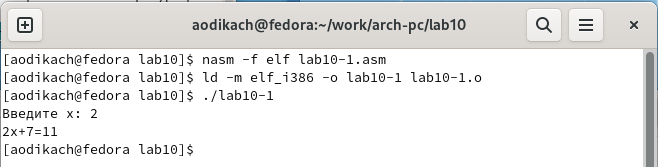


Рис. 1: пример работы исправленной программы

1. меняю текст программы, добавив подпрограмму \_subcalcul в подпрограмму \_calcul, для вычисления выражения 𝑓(𝑔(𝑥)), где 𝑥 вводится с клавиатуры, 𝑓(𝑥) = 2𝑥 + 7, 𝑔(𝑥) = 3𝑥 − 1 (рис. 2)

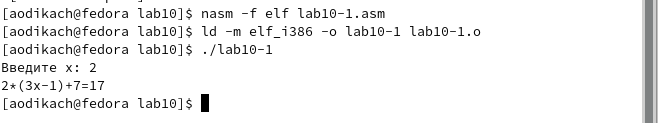


Рис. 2: работа программы для вычисления выражения

## 2.2 Отладка программ с помощью GDB

1. создаю файл lab10-2.asm с текстом программы из листинга 10.2, получаю исполняемый файл, добавляю отладочную информацию (рис. 3)

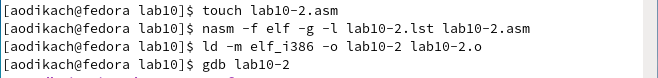


Рис. 3: добавление отладочной информации, загрузка файла в отладчик

1. проверяю работу программы запустив её в оболочке GDB с помощью команды run (рис. 4)

Рис. 4: проверка работы программы в оболочке GDB с помощью команды run

Рис. 4: проверка работы программы в оболочке GDB с помощью команды run

1. для более подробного анализа программы устанавливаю брейкпоинт на метку \_start, запускаю её (рис. 5)

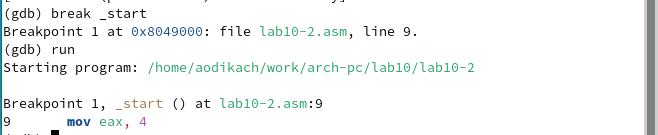


Рис. 5: пример работы программы

1. просматриваю дисассимилированный код программы с помощью команды disassemble (рис. 6)

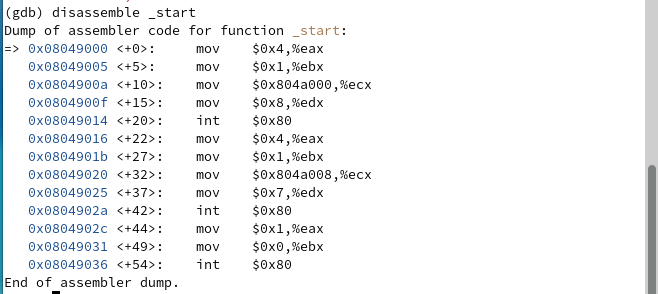
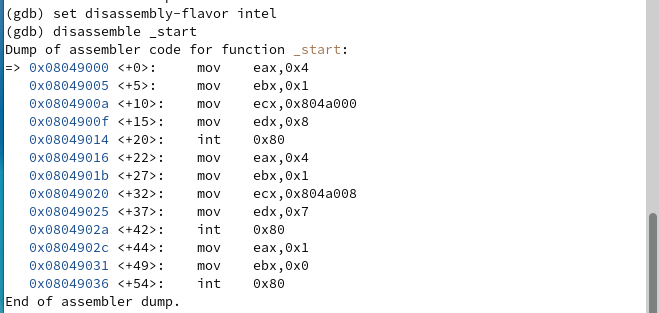


Рис. 6: просмотр дисассимилированного кода

1. переключаюсь на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel (рис. **¿fig:007?**). после смены интерфейса команды отображаются с привычным intel’ловским синтаксисом 

## 2.3 Добавление точек останова

1. проверяю была ли установлена точка останова с помощью команды info breakpoints (рис. 7)

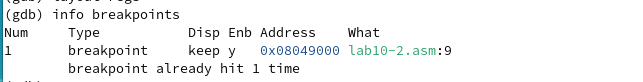


Рис. 7: проверка точки останова

1. устанавливаю ещё одну точку останова по адресу инструкции с помощью команды break \*
2. просматриваю информацию о всех установленных точках останова (рис. 8)



Рис. 8: информация о всех установленных точках

## 2.4 Работа с данными программы в GDB

1. выполняю 5 инструкций с помощью команды stepi ,,,
2. просматриваю значение переменной msg1 по имени (рис. 9)

Рис. 9: значение переменной msg1

Рис. 9: значение переменной msg1

1. просматриваю значение переменной msg2 по адресу (рис. 10)

Рис. 10: значение переменной msg2

Рис. 10: значение переменной msg2

1. просматриваю инструкцию mov ecx,msg2 которая записывает в регистр ecx адрес перемененной msg2
2. изменяю первый символ переменной msg1 (рис. 11)

Рис. 11: изменение первого символа переменной msg1

Рис. 11: изменение первого символа переменной msg1

1. заменяю символ во второй переменной msg2 с помощью команды set {char}0x804a009=‘0’ (рис. 12)

Рис. 12: изменение второго символа переменной msg2

Рис. 12: изменение второго символа переменной msg2

1. вывожу в различных форматах значение регистра edx (рис. 13).

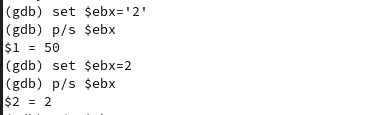


Рис. 13: вывод при разных значениях регистра

1. завершаю выполнение программы с помощью команды continue и выхожу из GDB с помощью команды quit (рис. 14).

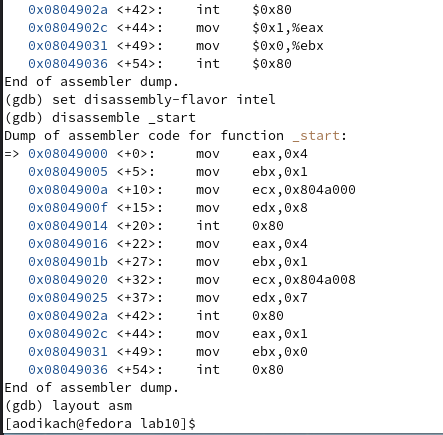


Рис. 14: вид после завершения

## 2.5 Обработка аргументов командной строки в GDB

1. копирую файл lab9-2.asm, создаю исполняемый файл, загружаю файл в отладчик, указав аргументы (рис. 15).

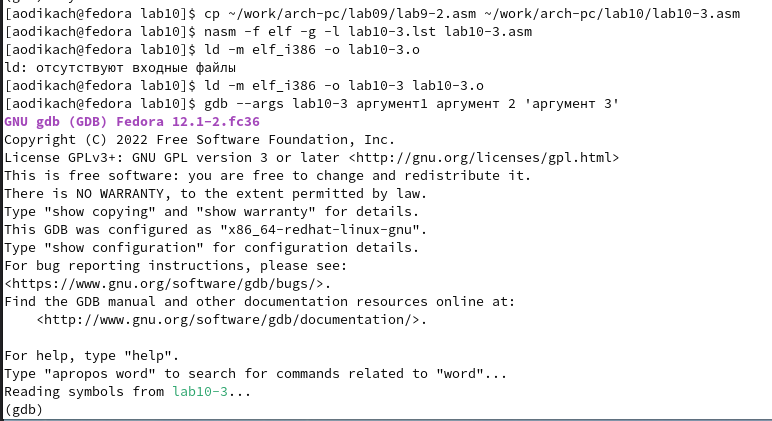


Рис. 15: копирование, создание исполняемого файла. загрузка файла в отладчик с указанием аргументов

1. устанавливаю точку останова перед первой инструкцией и запуска её (рис. 16).

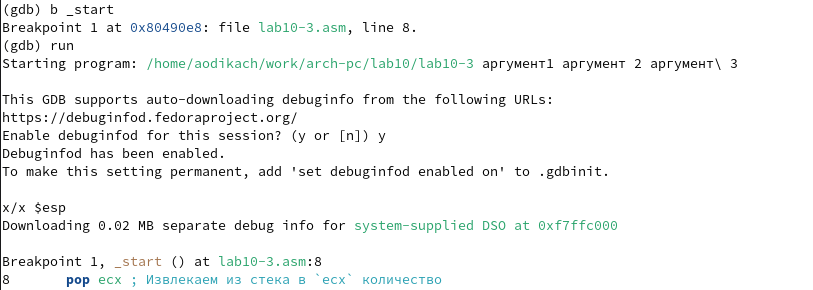


Рис. 16: пример работы

1. смотрю позиции стека по их адресам. шаг изменения равен 4 т.к размер переменной 4 байта (рис. 17).

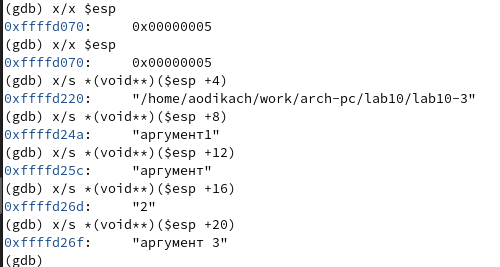
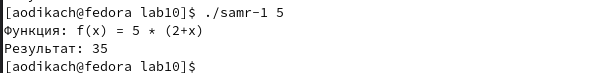


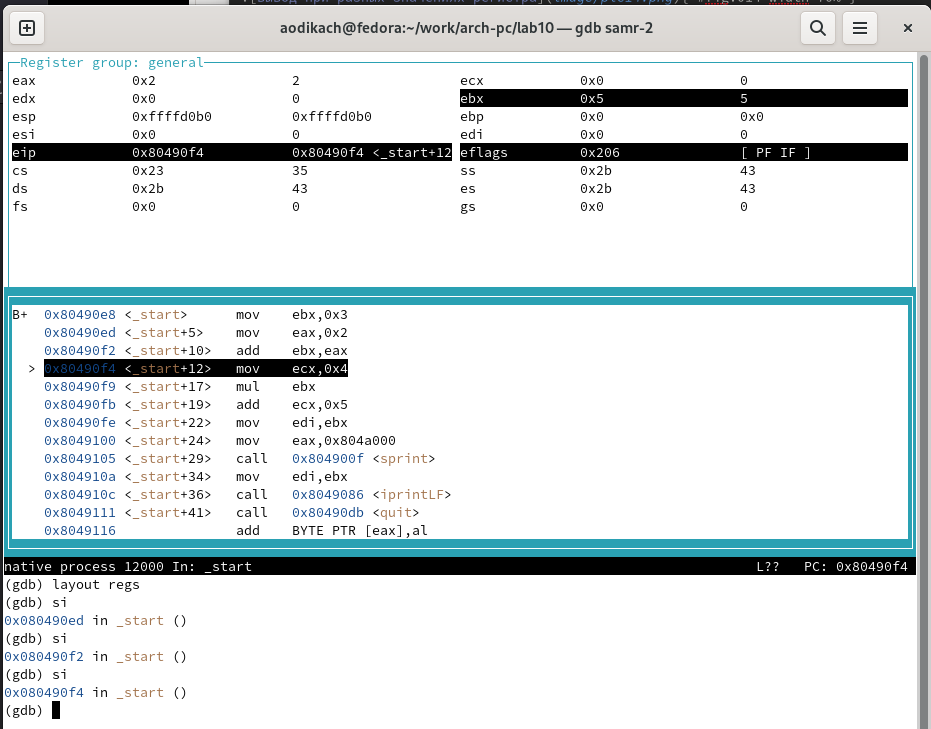
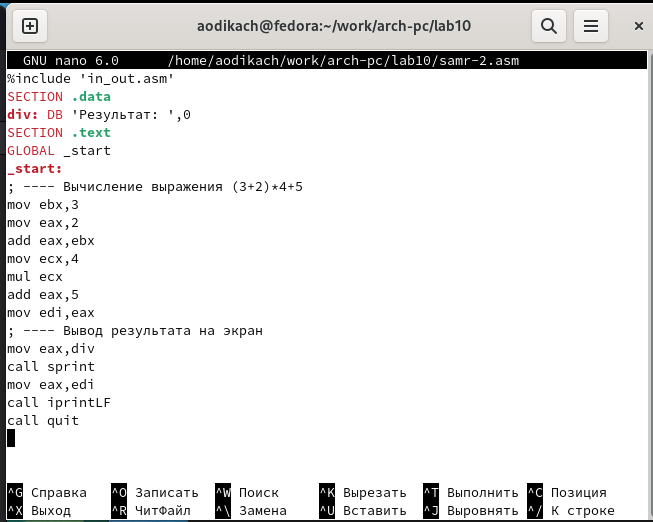
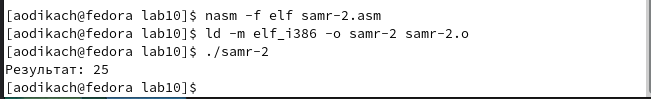
Рис. 17: позиции стека

## 2.6 Задание для самостоятельной работы

1. преобразование программы из лабораторной №9 с реализацией вычисления функции как подпрограммы (рис. **¿fig:019?**)(рис. **¿fig:020?**) (работаю в файле samr-1)

1. создаю файл samr-2, ввожу текст программы листинга, открываю откладчик, отлаживаю код (рис. **¿fig:021?**). в коде регистр edi копирует данные из еbx, а то время как копировать должен из eax, 5 должно прибавляться к eax, а также регистры ebx и eax перепутаны местами. (рис. **¿fig:022?**)(рис. **¿fig:023?**)

# 3 Вывод

научилась писать программы с использованием подпрограмм, а также попробовала переписать старую программу с использованием новой подпрограммы. узнала как пользоваться отладчиком и что такое отладка