ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №10

дисциплина: Архитектура компьютера

Шурыгин Илья Максимович

Содержание

# 1 Цель работы

Приобрети навык написания программ с использованием подпрограмм. Познакомиться с методами отладки при помощи GDB и его основными возможностями.

# 2 Задание

Необходимо написать программы, которые вычисляют значение функции, а затем проверить их работу в отладчике.

# 3 Выполнение лабораторной работы

1. Откроием Midnight Commander и перейдем в каталог ~/work/study. Создадим каталог для программам лабораторной работы N10, перейдем в него и создадим файл lab10-1.asm.(рис. 1)

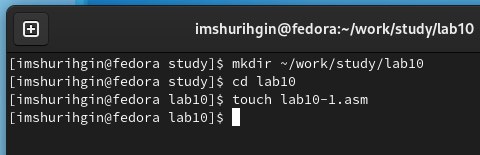


Рис. 1: Создали каталог lab10 в Midnight Commander и файл lab10-1.asm

1. Запишем в файл lab10-1.asm текст программы из листинга 10.1. Она вычисляет f(x) = 2x + 7 с помощью подпрограммы \_calcul. Изменим текст программы. Теперь программа вычисляет выражение f(g(x)), где f(x) = 2x + 7, g(x) = 3x − 1.(рис. 2)(рис. 3)(рис. 4)

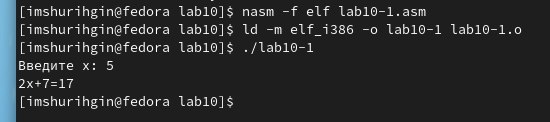


Рис. 2: Вычисляет f(x) = 2x + 7

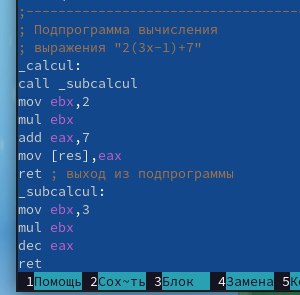


Рис. 3: Код измененной программы

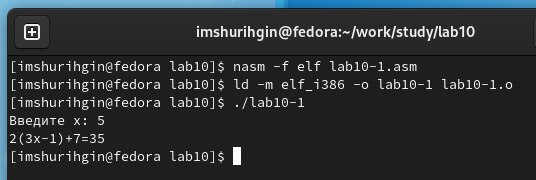


Рис. 4: Вычисляет f(g(x))

1. Создим файл lab10-2.asm с текстом программы из Листинга 10.2.(Программа печати сообщения Hello world!). Для работы с GDB в исполняемый файл добавим отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’.(рис. 5)

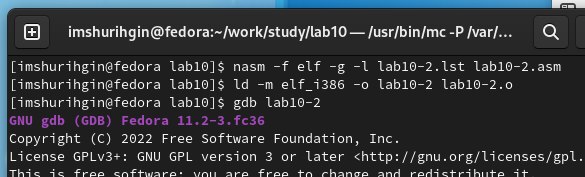


Рис. 5: Создание исполняемого файла

1. Запустим программу в оболочке GDB с помощью команды run.(рис. 6)

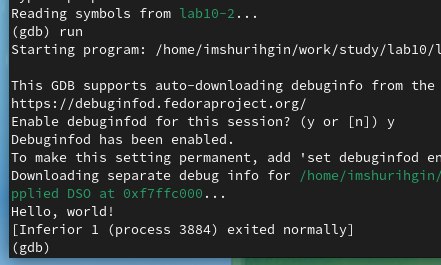


Рис. 6: Запуск программы lab10-2.asm

1. Установим брейкпоинт на метку \_start и посмотрим дисассимилированный код программы с помощью команды disassemble, начиная с метки \_start.(рис. 7)(рис. 8)

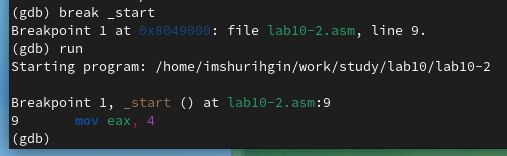


Рис. 7: Устанавливаем брейкпоинт

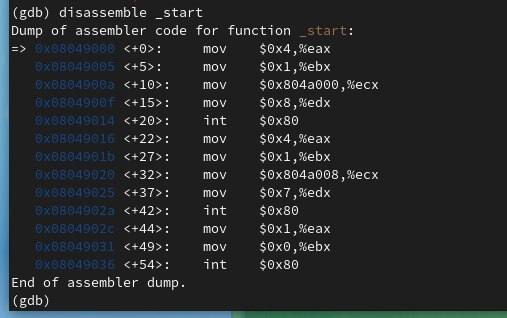


Рис. 8: Дисассимилированный код программы

1. Переключимся на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel.(рис. 9)

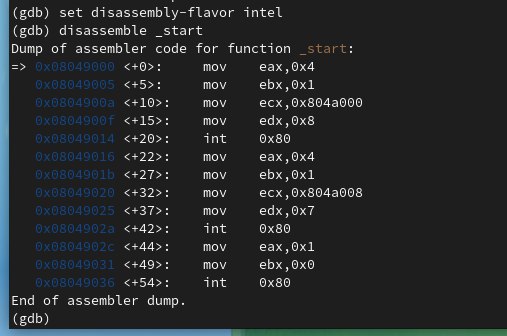


Рис. 9: Отображение команд с Intel’овским синтаксисом

1. Включим режим псевдографики для более удобного анализа программы и проверим, что была установлена точка останова(\_start) с помощью команды info breakpoints.(рис. 10)

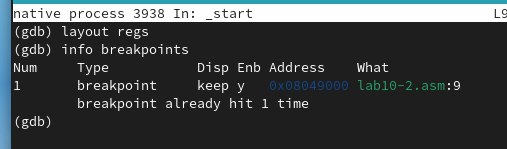


Рис. 10: Проверяем брейкпоинт

1. Установим еще одну точку останова по адресу инструкции. Определим адрес предпоследней инструкции (mov ebx,0x0) и установим точку останова. (рис. 11)(рис. 12)

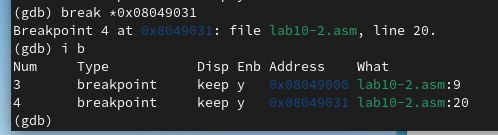


Рис. 11: Устанавливаем брейкпоинт

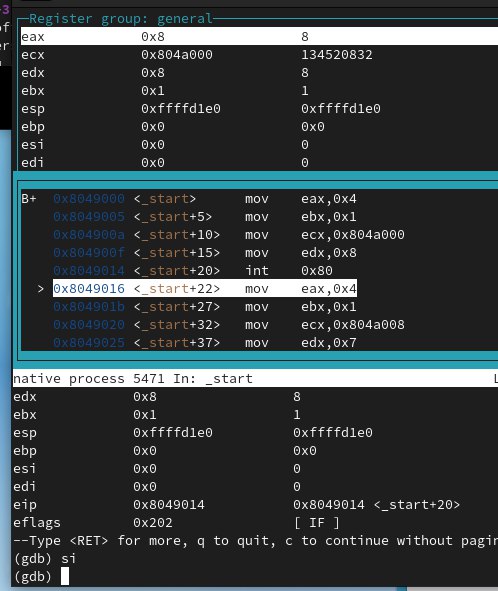


Рис. 12: Адрес предпоследней инструкции

1. Посмотрим значение переменной msg1 по имени, а значение переменной msg2 по адресу. Затем изменим первый символ переменной msg1 и любой символ во второй переменной msg2.(рис. 13)(рис. 14)



Рис. 13: Значения переменных msg1 и msg2



Рис. 14: Заменяем символы

1. Выведем в различных форматах значение регистра edx, а затем с помощью команды set изменим значение регистра ebx. Завершим выполнение программы с помощью команды continue или stepi и выйдем из GDB с помощью команды quit.(рис. 15)(рис. 16)

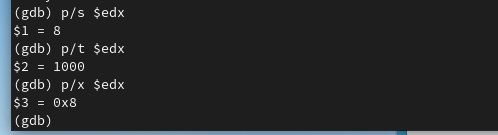


Рис. 15: Вывод значения регистра edx

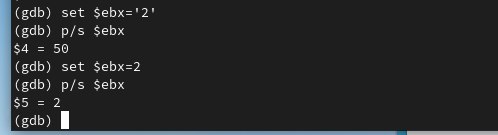


Рис. 16: Замена значения регистра ebx

1. Скопируем файл lab9-2.asm, созданный при выполнении лабораторной работы N9, создадим исполняемый файл и загрузим исполняемый файл в отладчик, указав аргументы.(рис. 17)

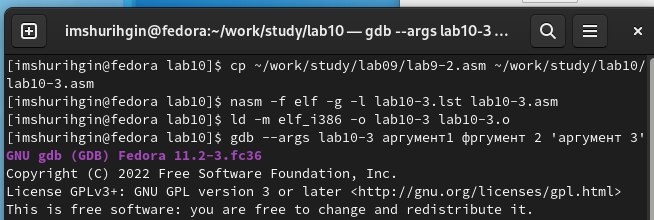


Рис. 17: Загрузка исполняемого файла в отладчик

1. Установим точку останова перед первой инструкцией в программе и запустим ее. Проверим число аргументов командной строки, которое распологается в регистре esp.(рис. 18)

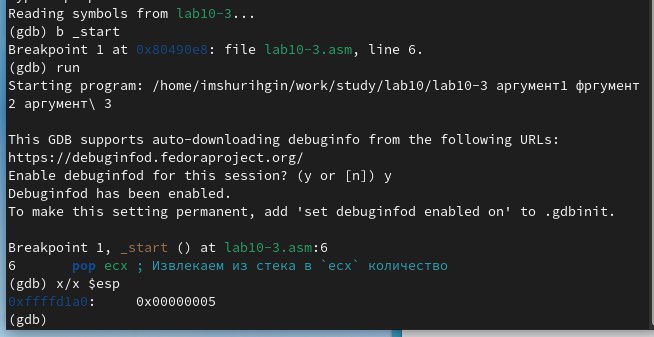


Рис. 18: Установили точку останова и проверили число аргументов

1. Посмотрим остальные позиции стека – по адесу (esp+4) располагается адрес в памяти, где находиться имя программы, по адесу (esp+8) храниться адрес первого аргумента и т.д.(рис. 19)

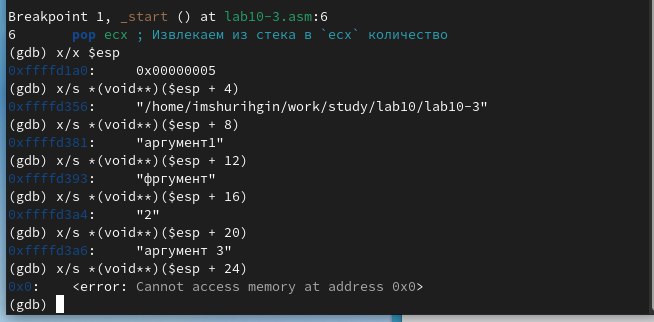


Рис. 19: Просмотр позиций стека

# 4 Задание для самостоятельной работы:

1. Преобразуем программу из лабораторной работы N9 (Задание N1 для самостоятельной работы), реализовав вычисление значения функции f(x) как подпрограмму.(рис. 20)(рис. 21)

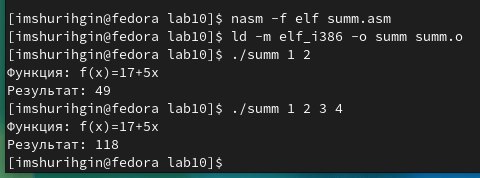


Рис. 20: Результат работы программы

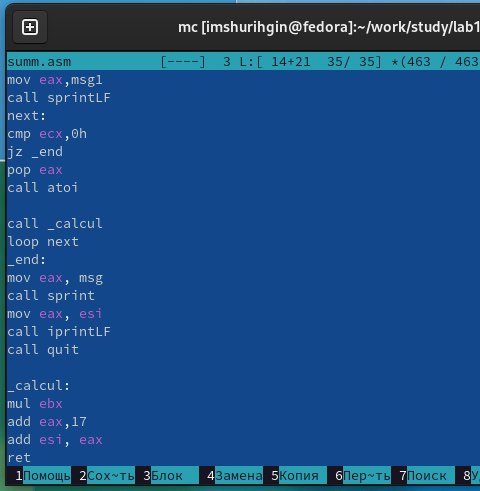


Рис. 21: Код программы

1. Нужно исправить работу программы, вычисляющей выражение: (3+2)\*4+5 с помощью отладчика GDB, анализируя изменения значений регистров. В отладчике видно, что программа умножает eax на 4, вместо того, чтобы умножать значение в регистре ebx, где лежит (3+2).(рис. 22)(рис. 23)(рис. 24)

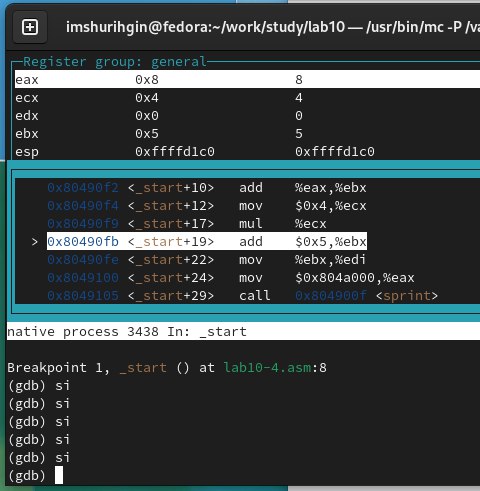


Рис. 22: Окно отладчика

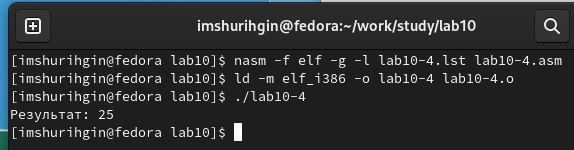


Рис. 23: Вывод программы

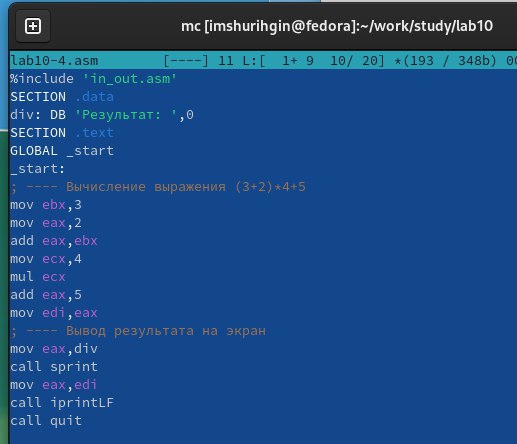


Рис. 24: Код программы

# 5 Выводы

В данной лабораторной работе я научился писать простые программы на языке ассемблера NASM, а именно: создал программу которая вычисляет значение функции с помощью подпрограммы. А также научился находить ошибки в коде программ с помощью отладчика GDB.