Лабораторная работа №9

Дисциплина: Архитектура компьютера

Серебрякова Дарья Ильинична

Содержание

# 1 Цель работы

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

# 2 Задания

1. Ознакомиться с понятием отладки
2. Ознакомиться со структурой подпрограмм
3. Научиться работать с подпрограммами и отладкой с помощью GDB

# 3 Теоретическое введение

## 3.1 Понятие об отладке

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа:

• обнаружение ошибки; • поиск её местонахождения; • определение причины ошибки; • исправление ошибки.

Можно выделить следующие типы ошибок:

• синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; • семантические ошибки — являются логическими и приводят к тому, что программа запускается, отрабатывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль).

Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга.

Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы.

Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново.

## 3.2 Методы отладки

Наиболее часто применяют следующие методы отладки:

• создание точек контроля значений на входе и выходе участка программы (например, вывод промежуточных значений на экран — так называемые диагностические сообщения); • использование специальных программ-отладчиков.

Отладчики позволяют управлять ходом выполнения программы, контролировать и изменять данные. Это помогает быстрее найти место ошибки в программе и ускорить её исправление. Наиболее популярные способы работы с отладчиком — это использование точек останова и выполнение программы по шагам. Пошаговое выполнение — это выполнение программы с остановкой после каждой строчки, чтобы программист мог проверить значения переменных и выполнить другие действия. Точки останова — это специально отмеченные места в программе, в которых программаотладчик приостанавливает выполнение программы и ждёт команд. Наиболее популярные виды точек останова:

• Breakpoint — точка останова (остановка происходит, когда выполнение доходит до определённой строки, адреса или процедуры, отмеченной программистом); • Watchpoint — точка просмотра (выполнение программы приостанавливается, если программа обратилась к определённой переменной: либо считала её значение, либо изменила его).

Точки останова устанавливаются в отладчике на время сеанса работы с кодом программы, т.е. они сохраняются до выхода из программы-отладчика или до смены отлаживаемой программы.

# 4 Выполнение лабораторной работы

## 4.1 Реализация подпрограмм в NASM

Создаю каталог для выполнения лабораторной работы № 9, перехожу в него и создаю файл lab09-1.asm (рис. 1).

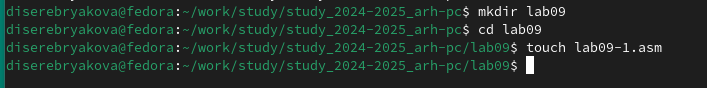


Рис. 1: 1

Изучаю текст программы из предложенного листинга и ввожу его в только что созданный файл (рис. 2).

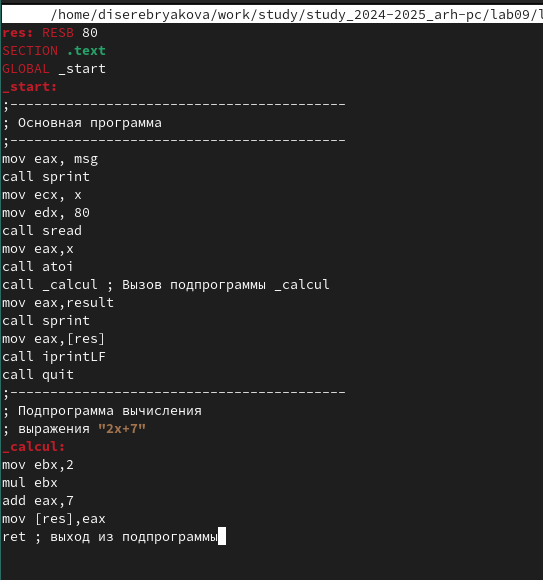


Рис. 2: 2

Создаю исполняемый файл и проверяю его работу (рис. 3).

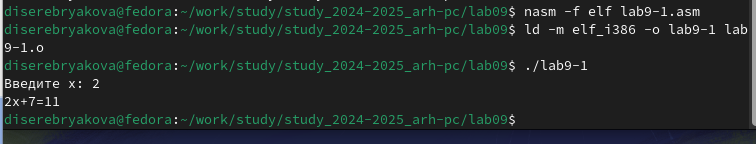


Рис. 3: 3

Значение функции при введенном с клавиатуры х посчитано верно. Изменяю 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. 4).

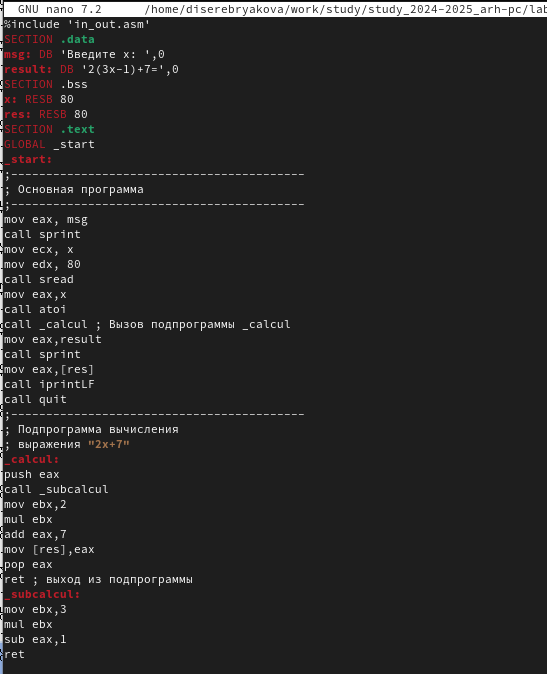


Рис. 4: 4

Создаю исполняемый файл и запускаю его (рис. 5).

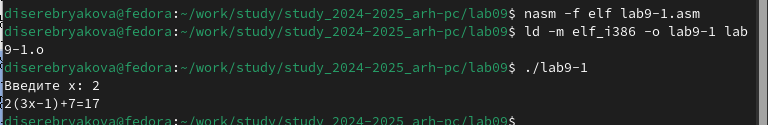


Рис. 5: 5

Значение подсчитано верно, значит программа написана правильно

## 4.2 Отладка программам с помощью GDB

Создаю файл lab09-2.asm и ввожу в него текст программы из предложенного листинга (рис. 6).

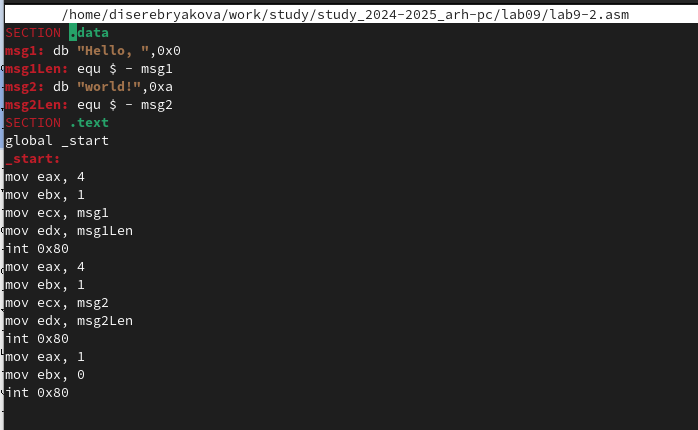


Рис. 6: 6

Создаю исполняемый файл, добавив отладочную информацию для работы с GDB. Проверяю работу программы, запустив ее в оболочке GDB с помощью команды run (рис. 7).

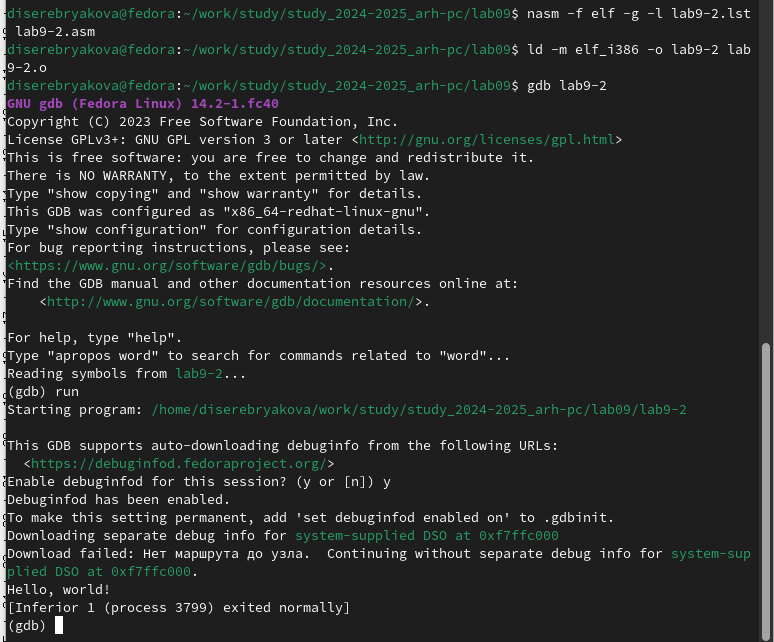


Рис. 7: 7

Для более подробного анализа программы устанавливаю брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запускаю её командой run (рис. 8).

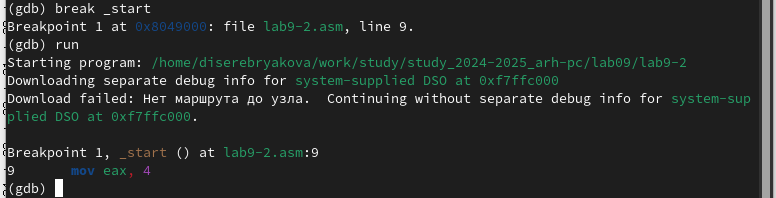


Рис. 8: 8

Далее просматриваю дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. 9).

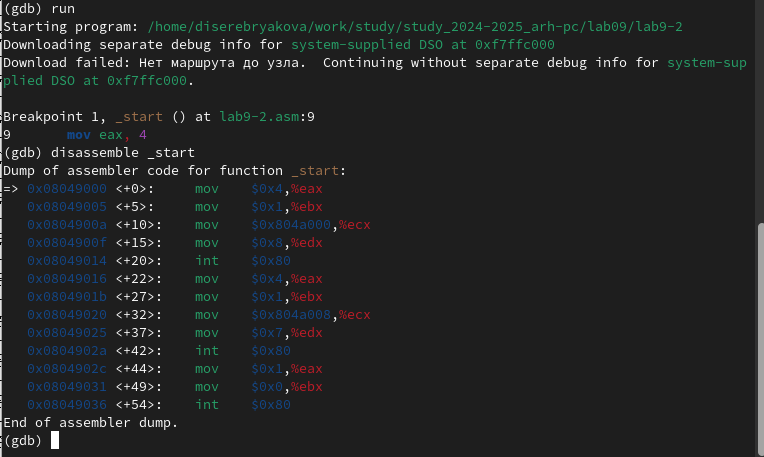


Рис. 9: 9

Переключаюсь на отображение команд с Intel’овским синтаксисом (рис. 10).

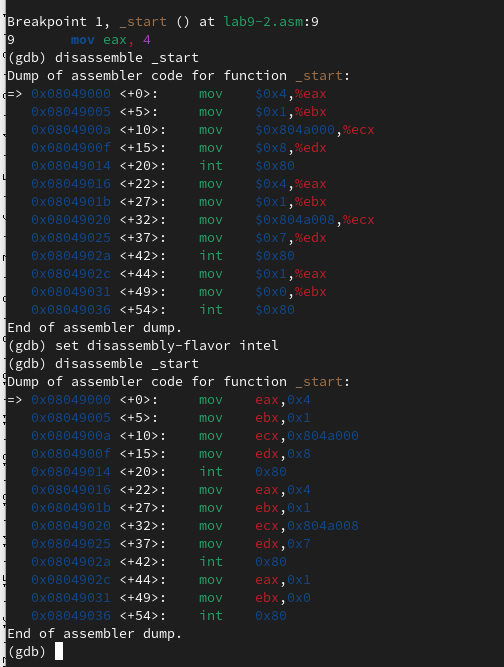


Рис. 10: 10

Различия между синтаксисом ATT и Intel заключаются в порядке операндов (ATT - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (ATT - pазмер операндов указывается явно с помощью суффиксов, непосредственные операнды предваряются символом $; Intel - Размер операндов неявно определяется контекстом, как ax, eax, непосредственные операнды пишутся напрямую), именах регистров(ATT - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

Включаю режим псевдографики для более удобного анализа программы (рис. 11).

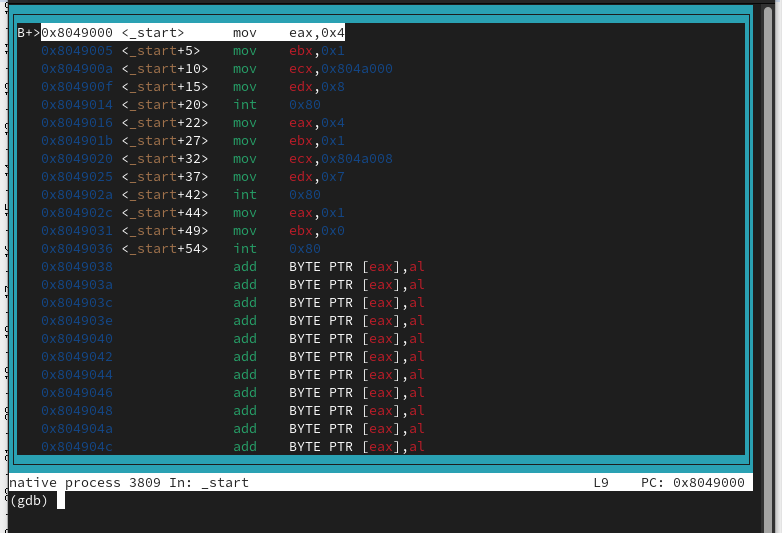


Рис. 11: 11

## 4.3 Добавление точек останова

На предыдущих шагах была установлена точка останова по имени метки (\_start). Проверяю это с помощью команды info breakpoints (рис. 12).

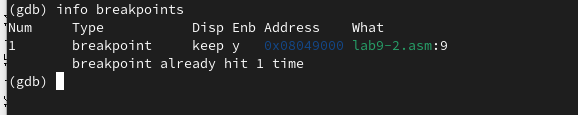


Рис. 12: 12

Устанавливаю еще одну точку останова по адресу предпоследней инструкции (mov ebx,0x0) и просматриваю информацию о всех установленных точках останова (рис. 13).

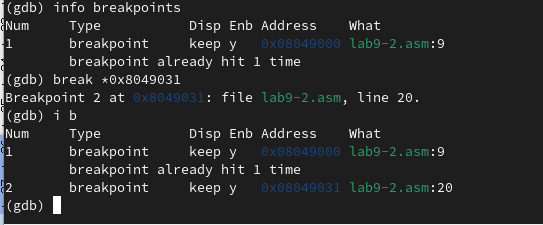


Рис. 13: 13

## 4.4 Работа с данными программы в GDB

Смотрю содержимое регистров с помощью команды info registers (рис. 14).

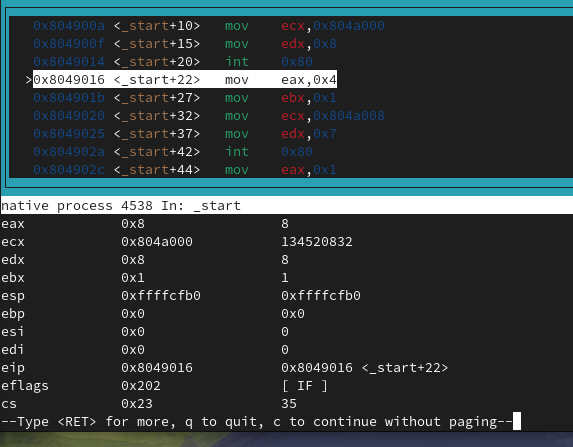


Рис. 14: 14

Смотрю значение переменной msg1 по имени (рис. 15).

Рис. 15: 15

Рис. 15: 15

Смотрю значение переменной msg2 по адресу (рис. 16).

Рис. 16: 16

Рис. 16: 16

Меняю первый символ переменной msg1 и первый символ переменной msg2 (рис. 17).

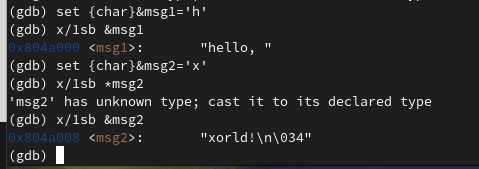


Рис. 17: 17

С помощью команды set изменяю значение регистра ebx (рис. 18).

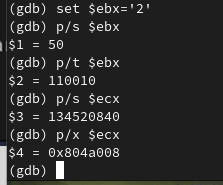


Рис. 18: 18

## 4.5 Обработка аргументов командной строки в GDB

Копирую файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки в файл с именем lab09-3.asm (рис. 19).

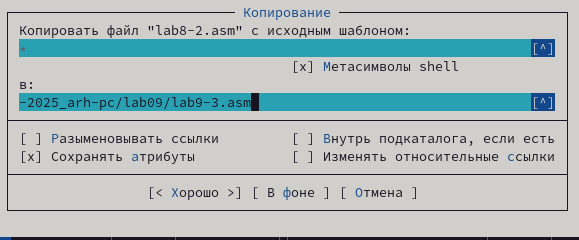


Рис. 19: 19

Создаю исполняемый файл и загружаю исполняемый файл в отладчик, указав аргументы (рис. 20).

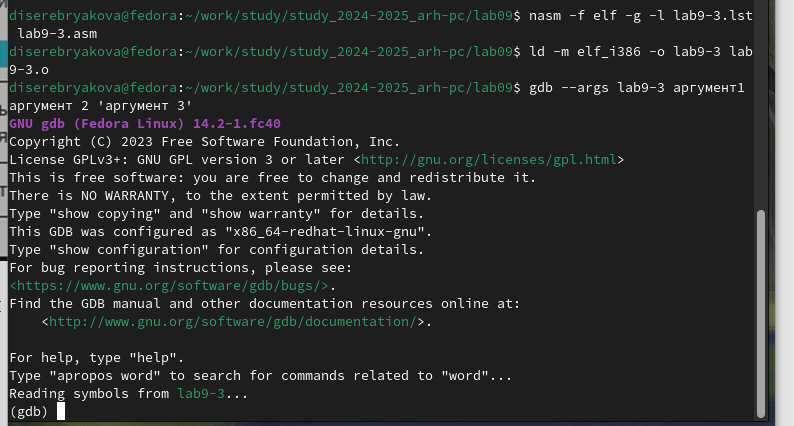


Рис. 20: 20

Исследую расположение аргументов командной строки в стеке после запуска программы с помощью gdb. Для начала устанавливаю точку останова перед первой инструкцией в программе и запускаю ее (рис. 21).

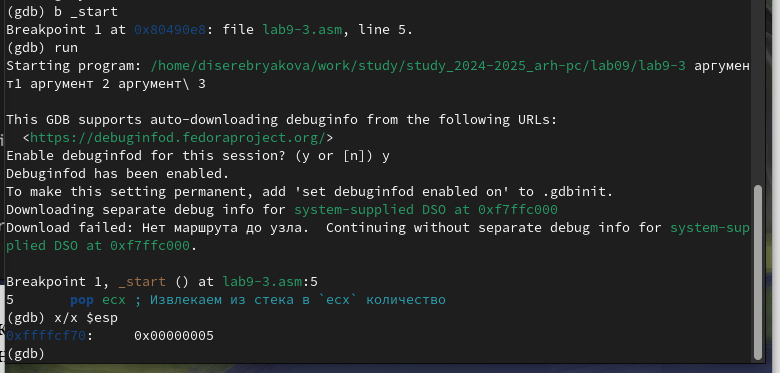


Рис. 21: 21

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы). Как видно, число аргументов равно 5 – это имя программы lab09-3 и непосредственно аргументы: аргумент1, аргумент, 2 и ‘аргумент 3’.

Просматриваю остальные позиции стека. По адесу [esp+4] располагается адрес в памяти где находиться имя программы, по адесу [esp+8] хранится адрес первого аргумента, по аресу [esp+12] – второго и т.д. (рис. 22).

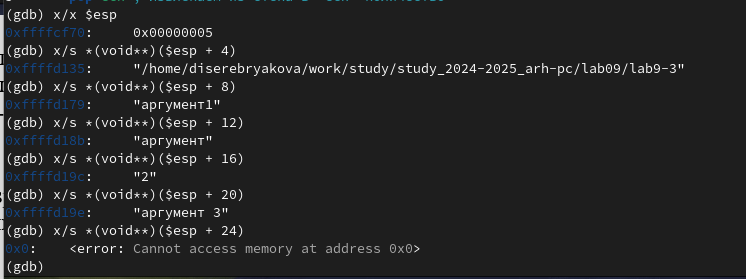


Рис. 22: 22

## 4.6 Задания для самостоятельной работы

Создаю файл lab9-4.asm и ввожу в него программу из предложенного листинга (рис. 23).

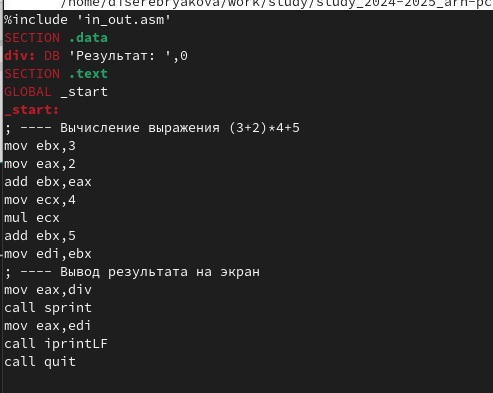


Рис. 23: 23

Запускаю программу в режике отладичка и пошагово через si просматриваю изменение значений регистров через i r. При выполнении инструкции mul ecx можно заметить, что результат умножения записывается в регистр eax, но также меняет и edx. Значение регистра ebx не обновляется напрямую, поэтому результат программы неверно подсчитывает функцию

Меняю код программы и запускаю ее повторно (рис. 24).

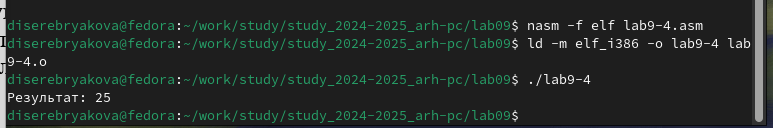


Рис. 24: 24

# 5 Вывод

В ходе выполнения работы приобретены навыки написания программ с использованием подпрограмм. Также ознакомилась с методами отладки при помощи GDB и его основными возможностями

# Список литературы

Лабораторная работа №9