Лабораторная работа №10

Понятие подпрограммы. Отладчик GDB

Медникова Екатерина Михайловна

Содержание

# 1 Цель работы

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

# 2 Теоретическое введение

**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.

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 программа-отладчик 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.

# 3 Выполнение лабораторной работы

1. Создала каталог для выполнения лабораторной работы No 10, перешла в него и создала файл lab10-1.asm.

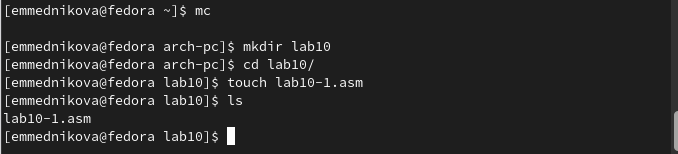


Figure 1: Создание каталога и файла

1. Ввела в файл lab10-1.asm текст программы из листинга 10.1.

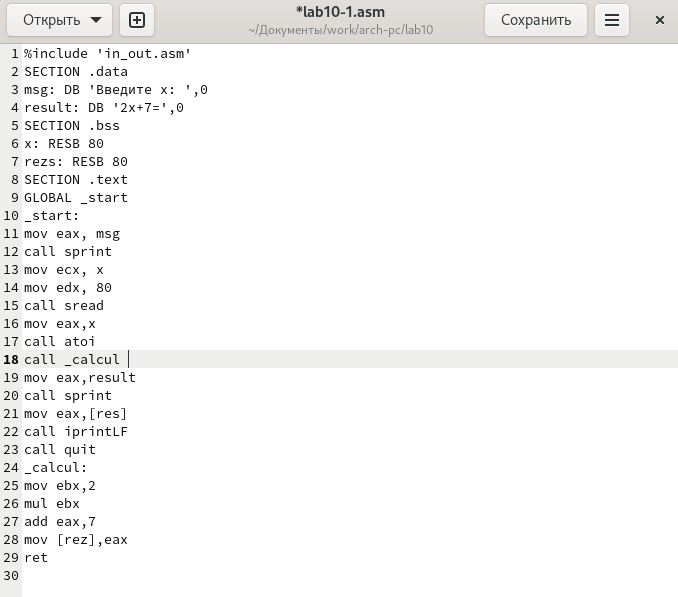


Figure 2: Ввод текста программы в файл

Создала исполняемый файл и проверила его работу. В результате программа не заработала, так как не распознаёт ‘res’ и ‘rez’.

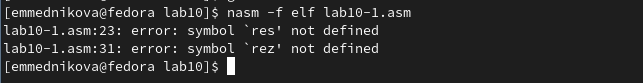


Figure 3: Создание и проверка работы файла

Изменила текст программы, добавив подпрограмму \_subcalcul в подпрограмму \_calcul.

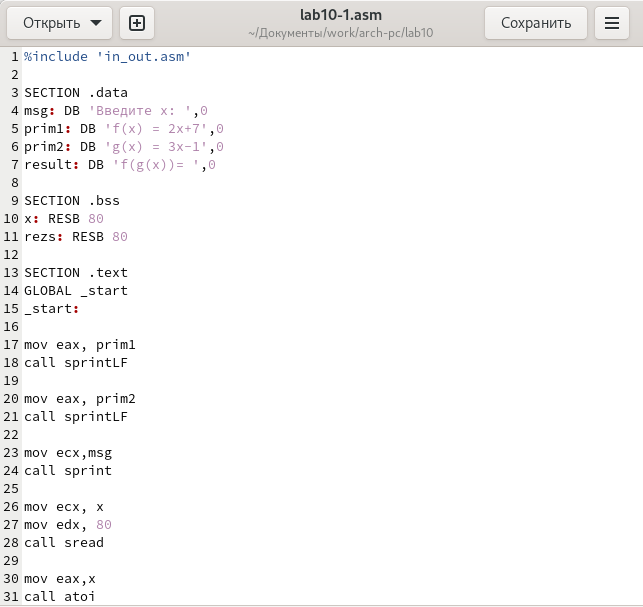


Figure 4: Изменение текста программы

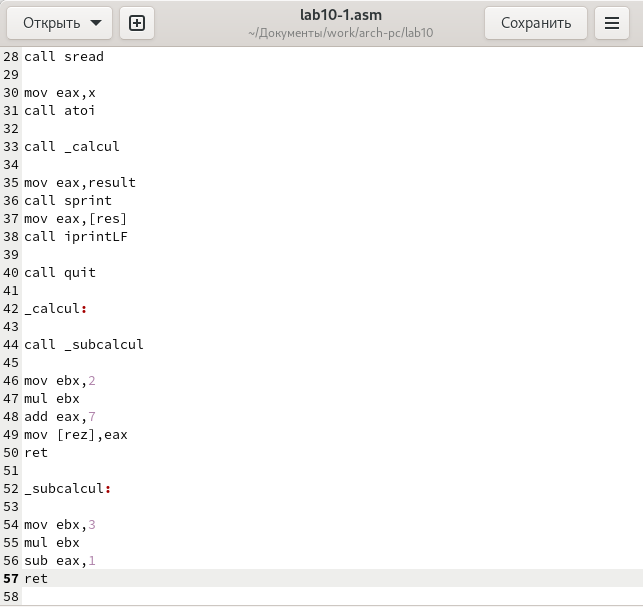


Figure 5: Изменение текста программы

При попытке создать исполняемый файл и проверить его работу, выдавалась та же самая ошибка, что и на рис. 3.3.

1. Создала файл lab10-2.asm с текстом программы из Листинга 10.2. (Программа печати сообщения Hello world!)

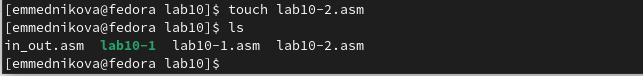


Figure 6: Создание файла

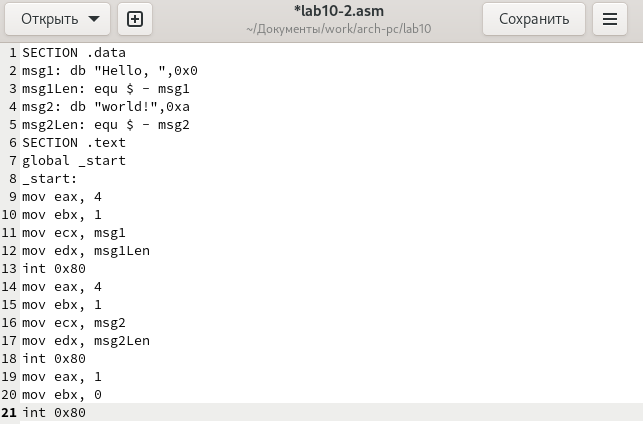


Figure 7: Ввод текста программы

Получила исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’. Загрузила исполняемый файл в отладчик gdb.

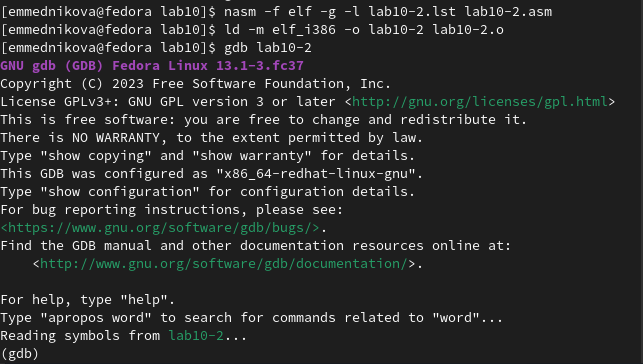


Figure 8: Создание исполняемого файла и загрузка в отладчик

Проверила работу программы, запустив ее в оболочке GDB с помощью команды run.

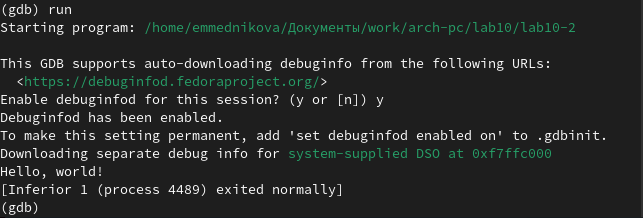


Figure 9: Проверка работы программы

Для более подробного анализа программы установила брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запустила её.

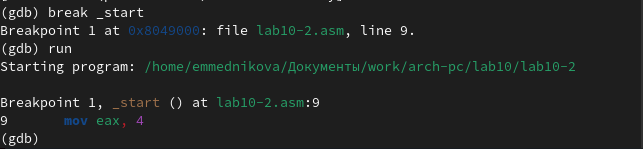


Figure 10: Анализ программы

Посмотрела дисассимилированный код программы с помощью команды disassemble начиная с метки \_start.

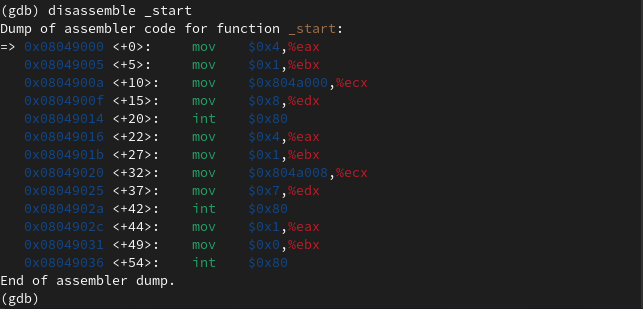


Figure 11: disassemble

Переключилась на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel.



Figure 12: set disassembly-flavor intel

Различия отображения синтаксиса машинных команд в режимах ATT и Intel: В коде Intel отсутствуют суффиксы обозначения размера; опускается символ % перед именами регистров; имеет другой способ описания местоположений в памяти.

Включила режим псевдографики для более удобного анализа программы.

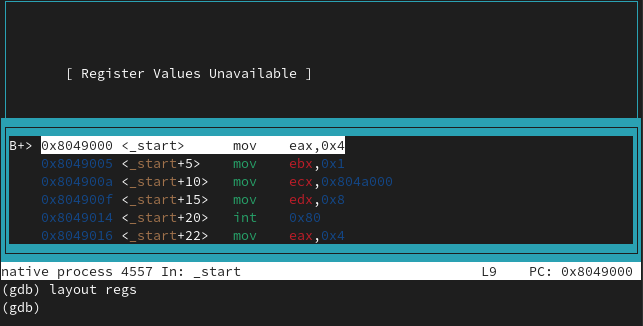


Figure 13: Режим псевдографики

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

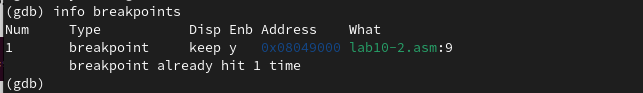


Figure 14: Точка останова

Определила адрес предпоследней инструкции (mov ebx,0x0) и установила точку останова.

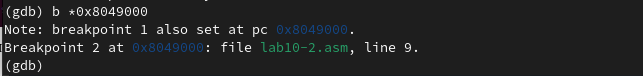


Figure 15: Установка точки останова

Посмотрела информацию о всех установленных точках останова.

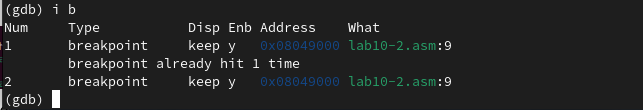


Figure 16: Информация о точках останова

1. Посмотрела содержимое регистров с помощью команды info registers.

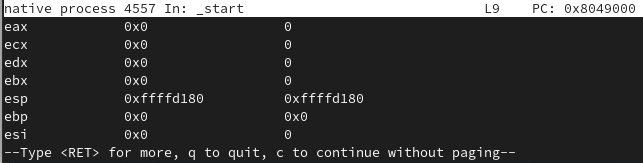


Figure 17: Содержимое регистров

Посмотрела значение переменной msg1 по имени.

Figure 18: Значение msg1

Figure 18: Значение msg1

Посмотрела значение переменной msg2 по адресу.

Figure 19: Значение msg2

Figure 19: Значение msg2

Изменила первый символ переменной msg1.

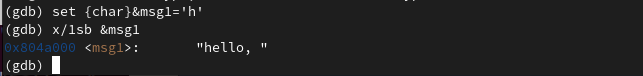


Figure 20: Изменение символа в msg1

Заменила символ во второй переменной msg2.

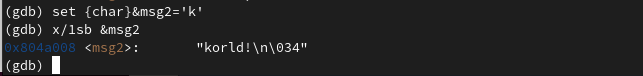


Figure 21: Изменение символа в msg2

Вывела в в шестнадцатеричном, в двоичном формате и в символьном виде значение регистра edx.

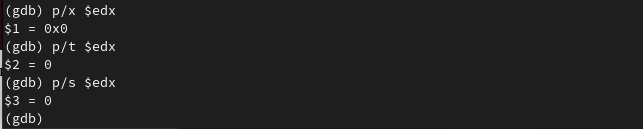


Figure 22: Значение регистра edx в разных форматах

С помощью команды set изменила значение регистра ebx.

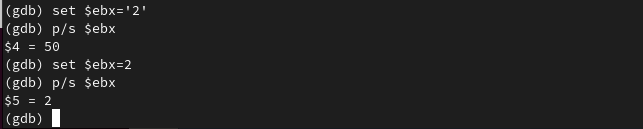


Figure 23: Изменение значения регистра ebx

1. Скопировала файл lab9-2.asm, созданный при выполнении лабораторной работы No9, с программой, выводящей на экран аргументы командной строки (Листинг 9.2) в файл с именем lab10-3.asm.

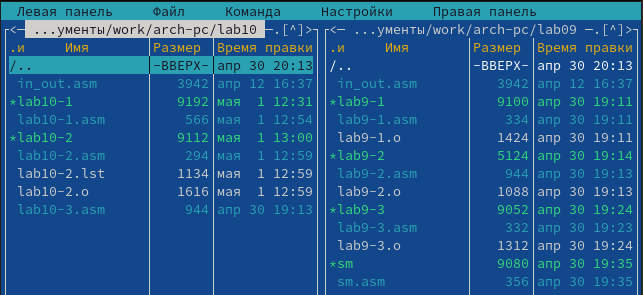


Figure 24: Копирование lab9-2.asm в lab10-3.asm

Создала исполняемый файл.

Figure 25: Создание исполняемого файла

Figure 25: Создание исполняемого файла

Загрузила исполняемый файл в отладчик, указав аргументы.

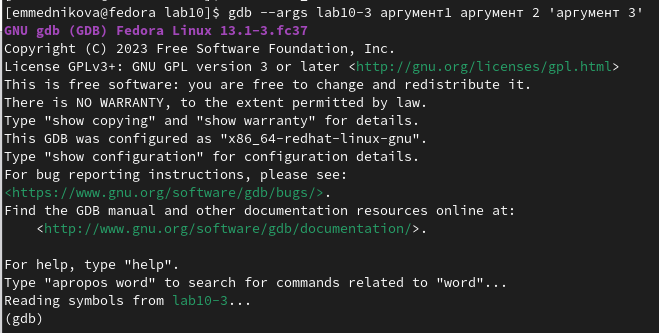


Figure 26: Загрузка в отладчик

Установила точку останова перед первой инструкцией в программе и запустила ее.

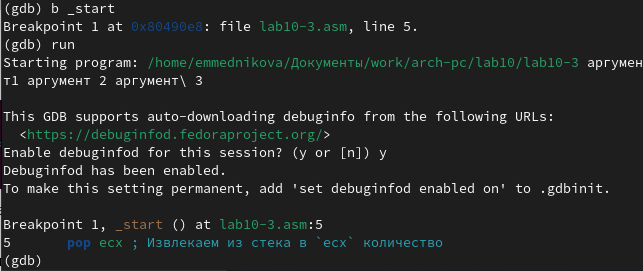


Figure 27: Установка точки останова

Адрес вершины стека хранится в регистре esp, и по этому адресу располагается число, равное количеству аргументов командной строки (включая имя программы).

Figure 28: Адрес вершины стека

Figure 28: Адрес вершины стека

Посмотрела остальные позиции стека – по адесу [esp+4] располагается адрес в памяти, где находится имя программы, по адесу [esp+8] хранится адрес первого аргумента, по аресу [esp+12] – второго и т.д.

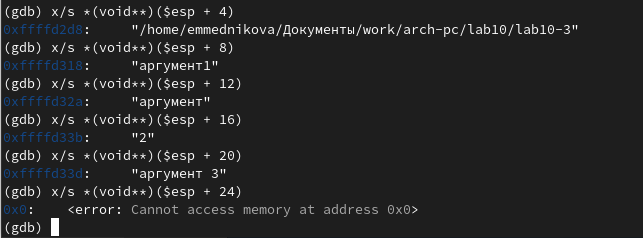


Figure 29: Позиции стека

# 4 Самостоятельная работа

1. Преобразовала программу из лабораторной работы No9 (Задание No1 для самостоятельной работы), реализовав вычисление значения функции как подпрограмму.

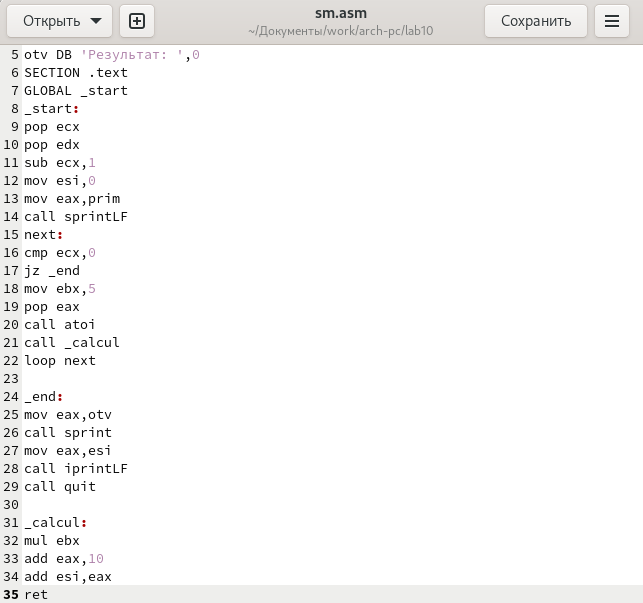


Figure 30: Преобразование программы

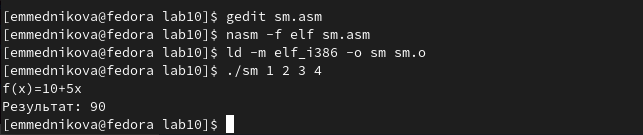


Figure 31: Проверка работы программы

1. С помощью отладчика GDB, анализируя изменения значений регистров, определила ошибку программы и исправила ее.

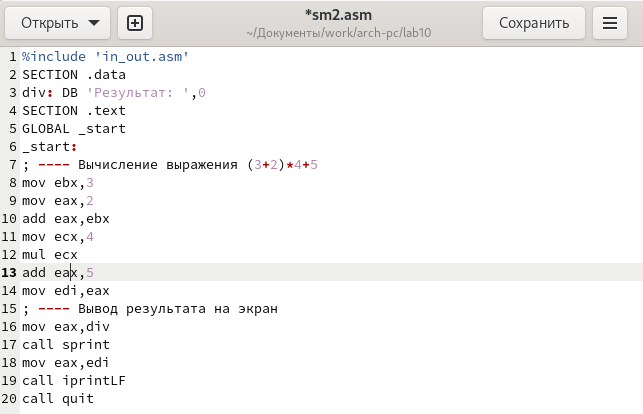


Figure 32: Исправление ошибки

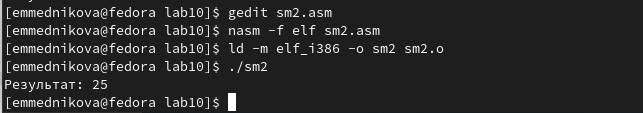


Figure 33: Проверка работы программы

# 5 Выводы

Приобрела навыки написания программ с использованием подпрограмм. Познакомилась с методами отладки при помощи GDB и его основными возможностями.