Отчёт по лабораторной работе №10

Понятие подпрограммы. Отладчик GDB.

Кочина Дарья Сергеевна

Содержание

# 1 Цель работы

Целью данной лабораторной работы является приобретение навыков написания программ с использованием подпрограмм. А также знакомство с методами отладки при помощи GDB и его основными возможностями.

# 2 Задание

Приобрести навыки написания программ с использованием подпрограмм. А также ознакомиться с методами отладки при помощи GDB и его основными возможностями.

# 3 Теоретическое введение

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

1. обнаружение ошибки;
2. поиск её местонахождения;
3. определение причины ошибки;
4. 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.

Наиболее часто применяют следующие методы отладки:

1. Создание точек контроля значений на входе и выходе участка программы;
2. Использование специальных программ-отладчиков.

**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:

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

GDB (GNU Debugger — отладчик проекта GNU) работает на многих UNIX-подобных системах и умеет производить отладку многих языков программирования.

GDB может выполнять следующие действия:

1. Начать выполнение программы, задав всё, что может повлиять на её поведение;
2. Остановить программу при указанных условиях;
3. Исследовать, что случилось, когда программа остановилась;
4. Изменить программу так, чтобы можно было поэкспериментировать с устранением эффектов одной ошибки и продолжить выявление других.

**Подпрограмма** — это, как правило, функционально законченный участок кода, который можно многократно вызывать из разных мест программы. В отличие от простых переходов из подпрограмм существует возврат на команду, следующую за вызовом.

# 4 Выполнение лабораторной работы

1. Я создала каталог для выполнения лабораторной работы №10, перешла в него и создала файл lab10-1.asm. (рис. 1)

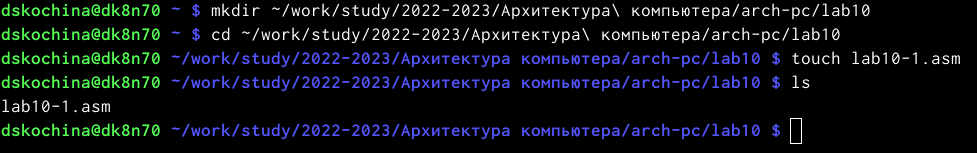


Рис. 1: Создание файла lab10-1.asm

1. Я ввела в файл lab10-1.asm текст программы из листинга (пример программы с использованием вызова подпрограммы), создала исполняемый файл и проверила его работу. (рис. 2, 3)

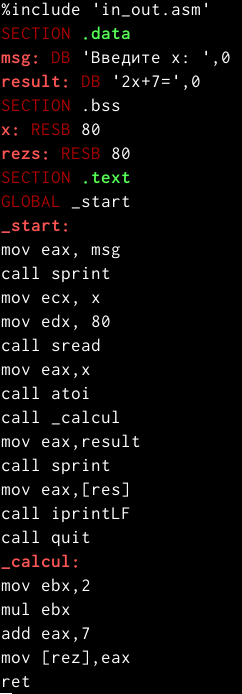


Рис. 2: Текст программы из листинга

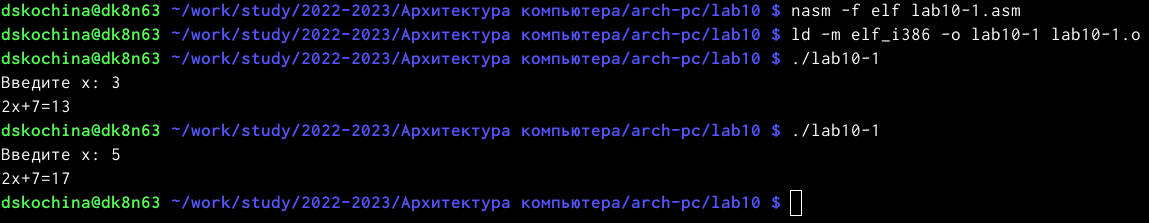


Рис. 3: Результат работы программы

1. Я изменила 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. Т.е. x передается в подпрограмму \_calcul из неё в подпрограмму \_subcalcul, где вычисляется выражение g(x), результат возвращается в \_calcul и вычисляется выражение f(g(x)). Результат возвращается в основную программу для вывода результата на экран. (рис. 4, 5)

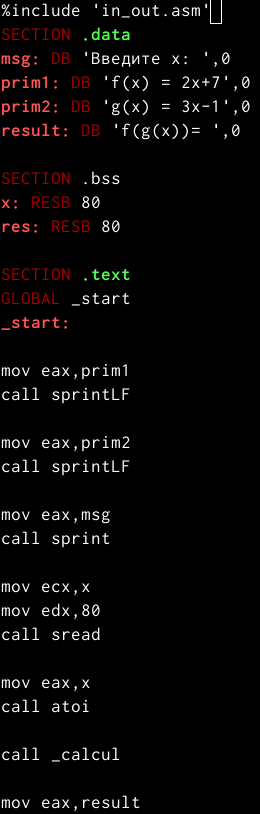


Рис. 4: Изменённый текст программы

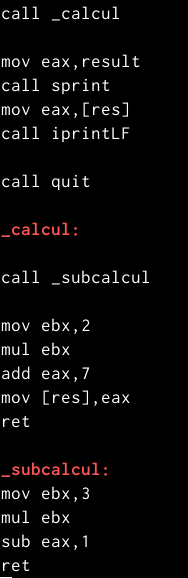


Рис. 5: Изменённый текст программы

1. Я создала исполняемый файл и проверила его работу. (рис. 6)

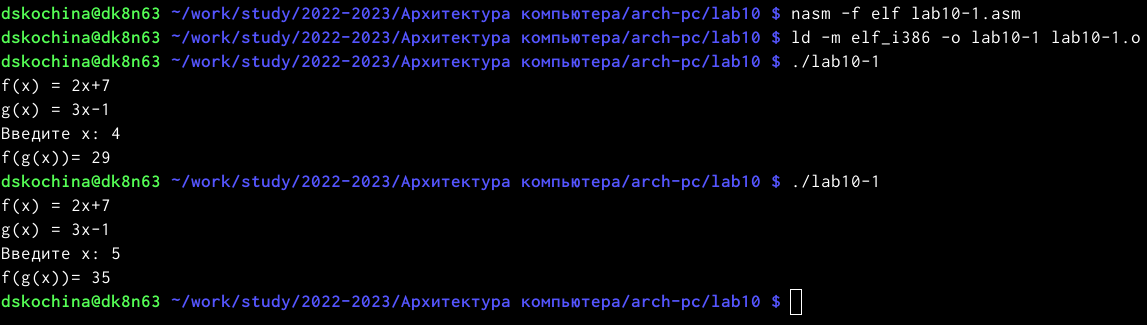


Рис. 6: Результат работы программы

1. Я создала файл lab10-2.asm с текстом программы из листинга (Программа печати сообщения Hello world!). (рис. 7, 8)

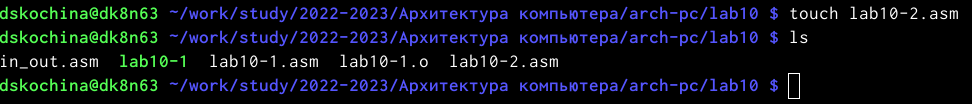


Рис. 7: Создание файла lab10-2.asm

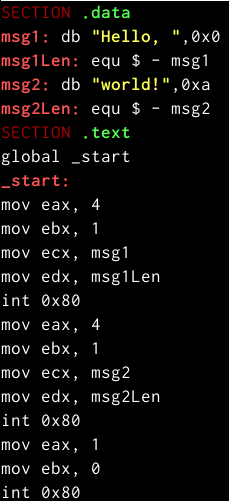


Рис. 8: Текст программы из листинга

1. Я получила исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’. Я загрузила исполняемый файл в отладчик gdb. (рис. 9)

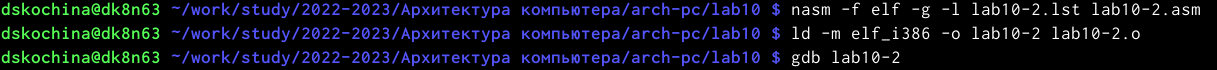


Рис. 9: Загрузка исполняемого файла в отладчик gdb

1. Я проверила работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r). (рис. 10)

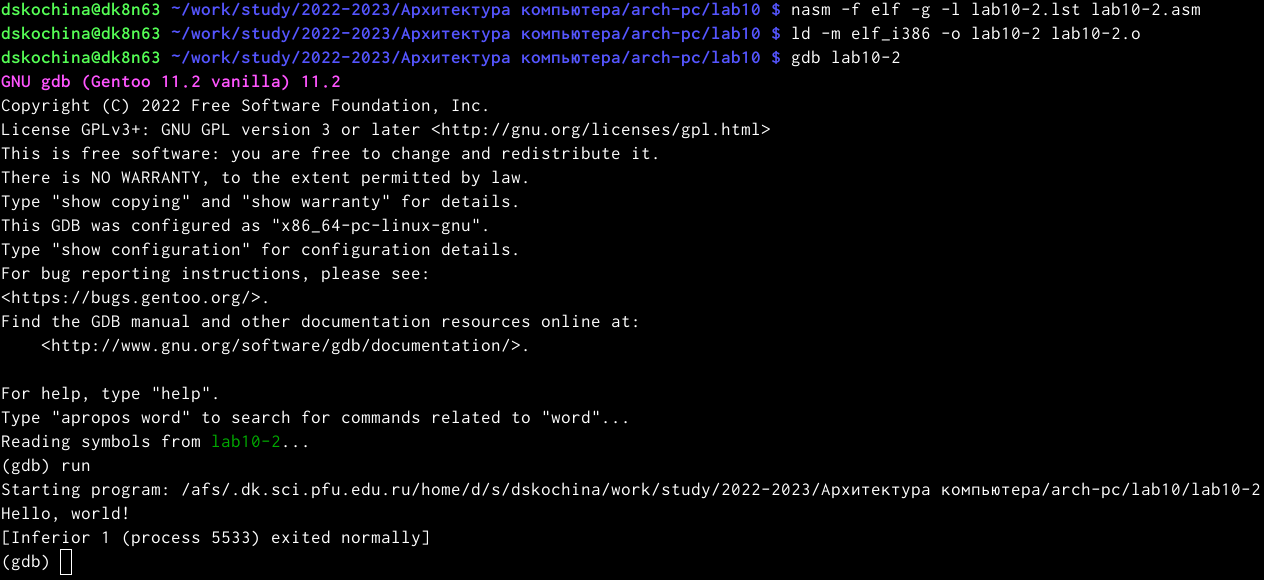


Рис. 10: Команда run

1. Для более подробного анализа программы я установила брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запустила её. (рис. 11)

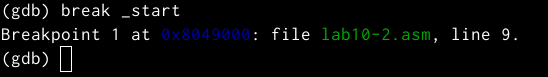


Рис. 11: Запуск программы

1. Я посмотрела дисассимилированный код программы с помощью команды disassemble начиная с метки \_start. (рис. 12, 13)

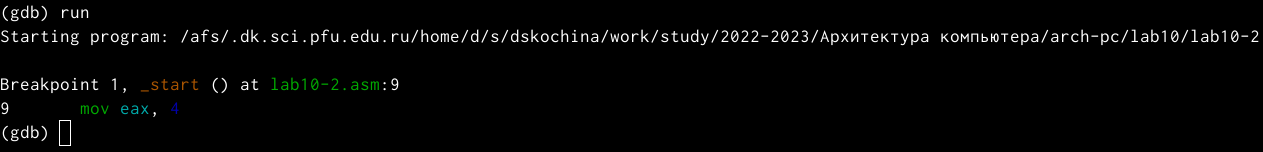


Рис. 12: Дисассимилированный код программы

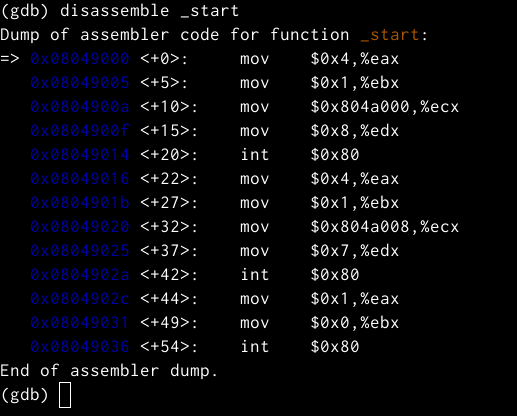


Рис. 13: Команда disassemble

1. Я переключилась на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel. (рис. 14)



Рис. 14: Команда set disassembly-flavor intel

1. Различия отображения синтаксиса машинных команд в режимах ATT и Intel.
2. Я включила режим псевдографики для более удобного анализа программы. (рис. 15, 16)

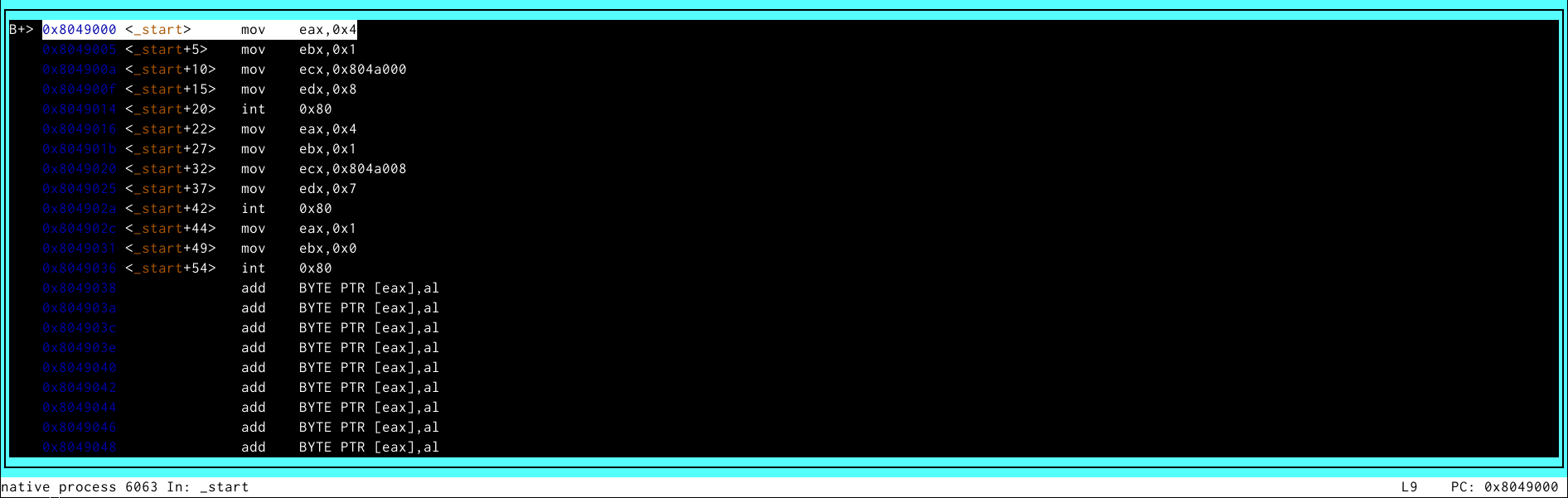


Рис. 15: Команда layout asm



Рис. 16: Команда layout regs

1. На предыдущих шагах была установлена точка останова по имени метки (\_start). Я проверила это с помощью команды info breakpoints (кратко i b). (рис. 17)

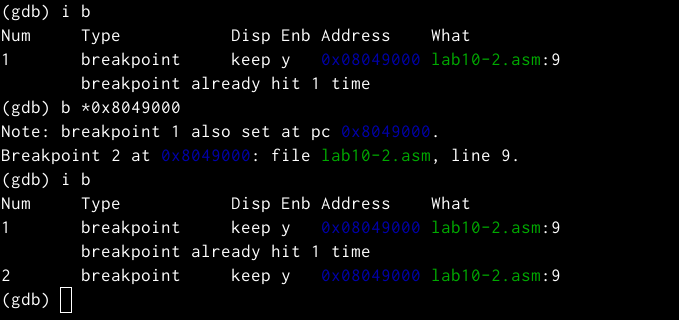


Рис. 17: Команда info breakpoints

1. Я установила ещё одну точку останова по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции. Я определила адрес предпоследней инструкции (mov ebx,0x0) и установила точку останова. (рис. 18)

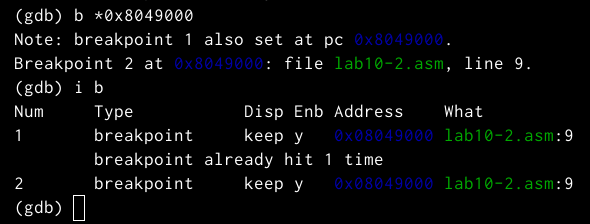


Рис. 18: Установка точки останова

1. Я выполнила 5 инструкций с помощью команды stepi (или si) и проследила за изменением значений регистров.
2. Я посмотрела содержимое регистров с помощью команды info registers (или i r).
3. Я посмотрела значение переменной msg1 по имени. (рис. 19)

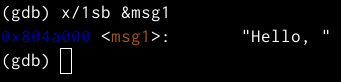


Рис. 19: Значение переменной msg1 по имени

1. Я посмотрела значение переменной msg2 по адресу. Адрес переменной можно определить по дисассемблированной инструкции. Также я посмотрела инструкцию mov ecx,msg2 которая записывает в регистр ecx адрес перемененной msg2.
2. Я изменила значение для регистра или ячейки памяти с помощью команды set. Также я изменила первый символ переменной msg1. (рис. 20, 21)

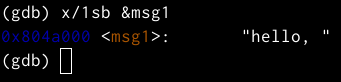


Рис. 20: Команда set

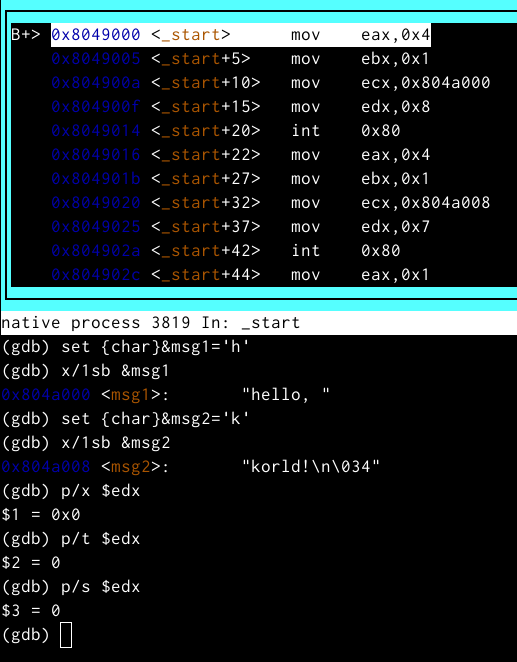


Рис. 21: Команда set

1. Я заменила символ во второй переменной msg2. (рис. 22)

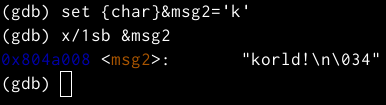


Рис. 22: Замена символа

1. Я вывела в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx.
2. С помощью команды set я изменила значение регистра ebx. (рис. 23, 24)

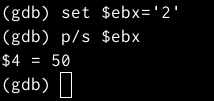


Рис. 23: Изменение значения регистра ebx

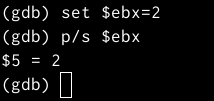


Рис. 24: Изменение значения регистра ebx

1. Разница вывода команд p/s $ebx.
2. Я завершила выполнение программы с помощью команды continue (сокращенно c) и вышла из GDB с помощью команды quit (сокращенно q). (рис. 25)

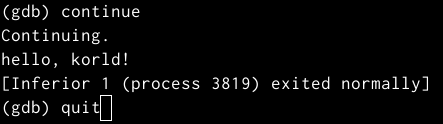


Рис. 25: Выход из GDB

1. Я скопировала файл lab9-2.asm, созданный при выполнении лабораторной работы №9, с программой, выводящей на экран аргументы командной строки в файл с именем lab10-3.asm. (рис. 26)

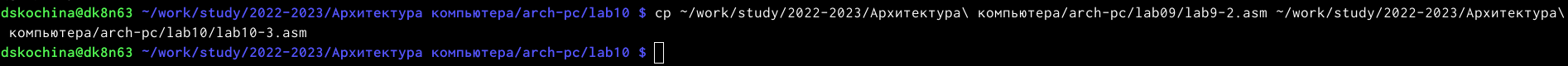


Рис. 26: Копирование

1. Я создала исполняемый файл. (рис. 27)

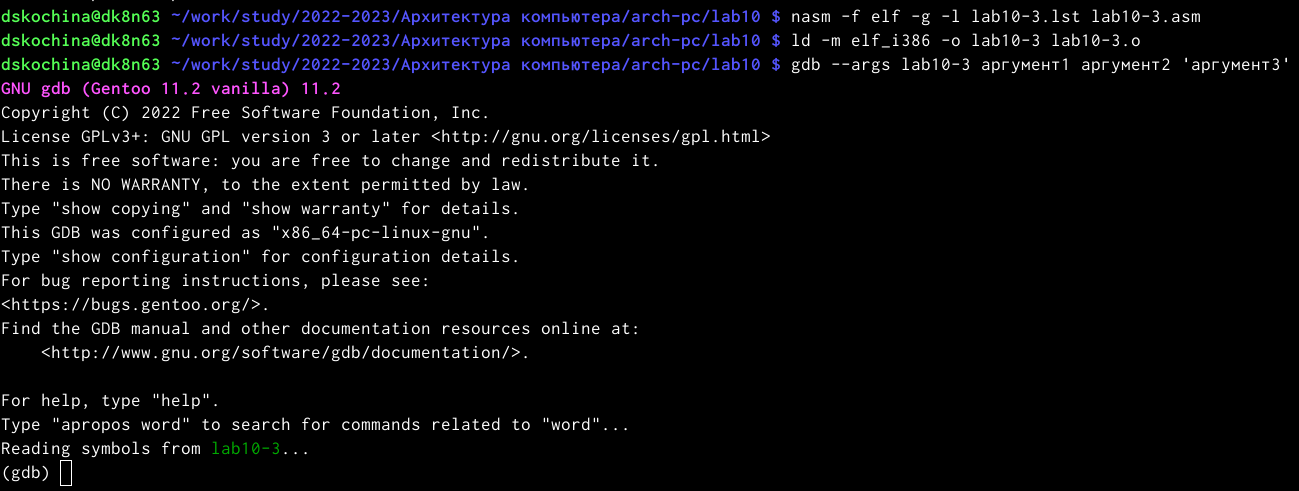


Рис. 27: Создание исполняемого файла

1. Для загрузки в gdb программы с аргументами необходимо использовать ключ –args. Я загрузила исполняемый файл в отладчик, указав аргументы.
2. Как отмечалось в предыдущей лабораторной работе, при запуске программы аргументы командной строки загружаются в стек. Исследуем расположение аргументов командной строки в стеке после запуска программы с помощью gdb. Для начала установим точку останова перед первой инструкцией в программе и запустим её. (рис. 28)

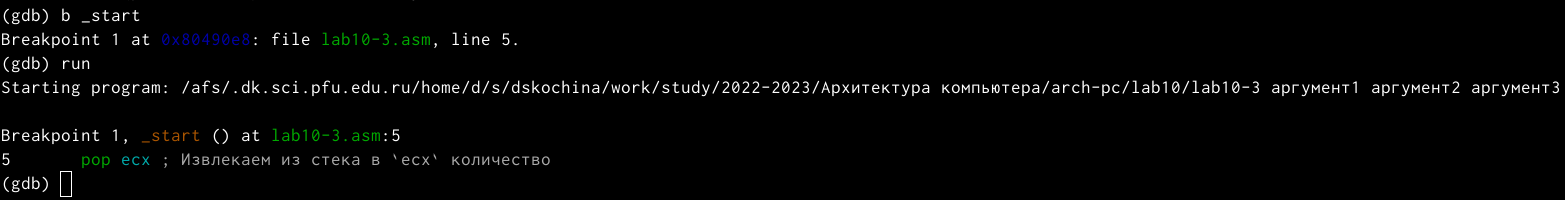


Рис. 28: Установка и запуск точки останова

1. Адрес вершины стека хранится в регистре esp и по этому адресу располагается число, равное количеству аргументов командной строки (включая имя программы). (рис. 29)

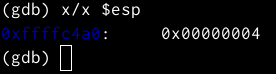


Рис. 29: Адрес вершины стека

1. Как видно, число аргументов равно 4 – это имя программы lab10-3 и непосредственно аргументы: аргумент1, аргумент, 2 и ‘аргумент 3’. Я посмотрела остальные позиции стека. (рис. 30)

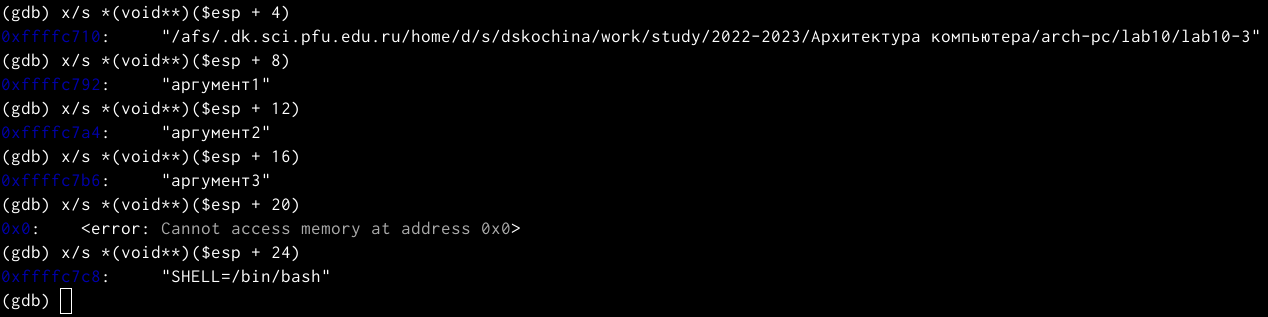


Рис. 30: Просмотр позиций стека

**Самостоятельная работа**

1. Я преобразовала программу из лабораторной работы №9 (задание №1 для самостоятельной работы), реализовав вычисление значения функции f(x) как подпрограмму.

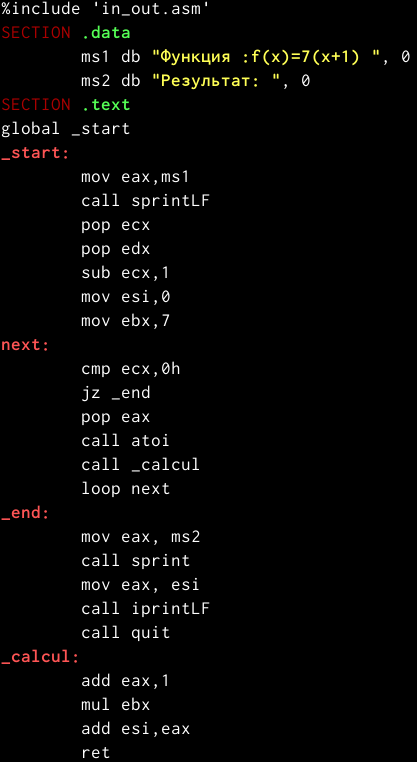


Рис. 31: Преобразование программы

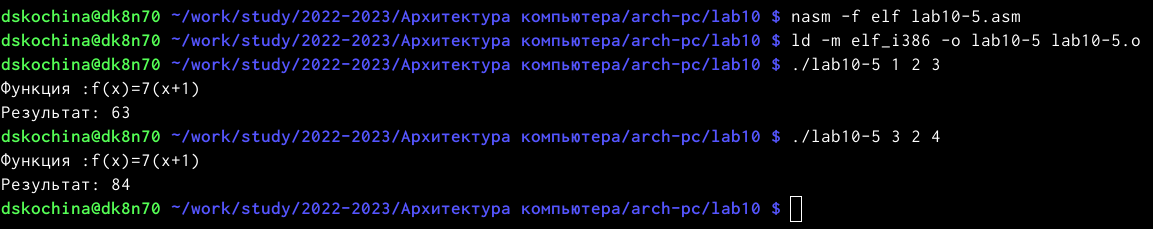


Рис. 32: Проверка работы программы

1. В листинге приведена программа вычисления выражения (3+2)\*4+5. При запуске данная программа даёт неверный результат. С помощью отладчика GDB, анализируя изменения значений регистров, я определила ошибку и исправила её. (рис. 33, 34, 35, 36)

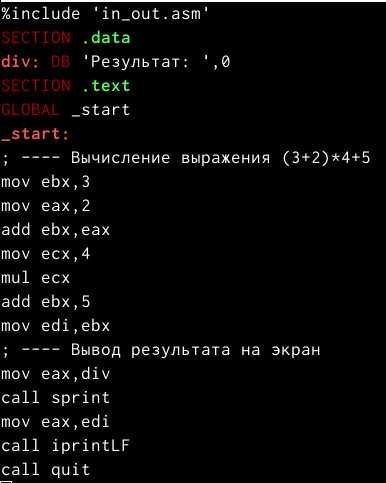


Рис. 33: Текст программы из листинга

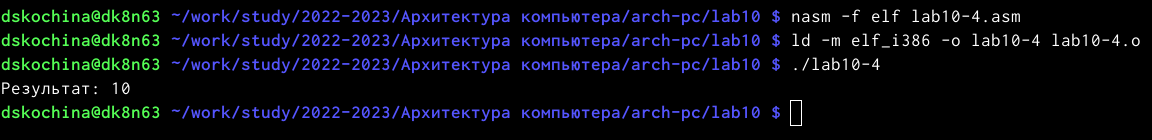


Рис. 34: Проверка программы

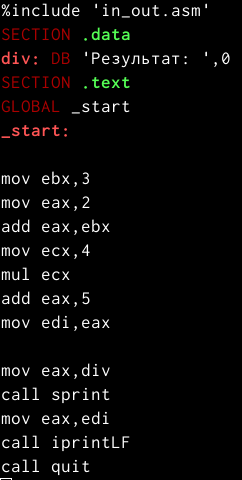


Рис. 35: Изменённый текст программы

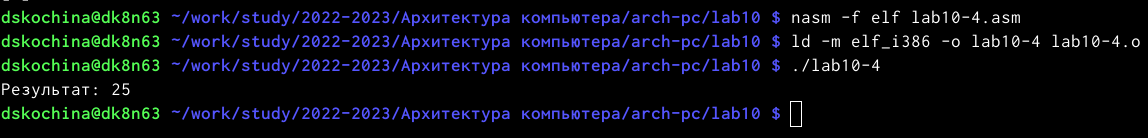


Рис. 36: Результат работы программы

# 5 Выводы

В ходе выполнения данной лабораторной работы я приобрела навыки написания программ с использованием подпрограмм. Ознакомилась с методами отладки при помощи GDB и его основными возможностями.