Лабораторная работа №9

Понятие подпрограммы. Отладчик GDB.

Седохин Даниил Алексеевич

Содержание

# 1 Цель работы

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

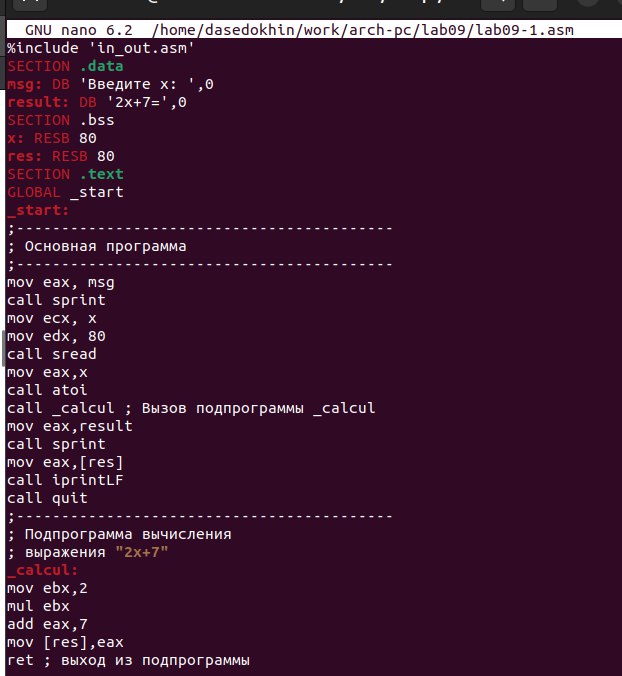
# 2 Выполнение лабораторной работы

1. ССоздадим каталог для выполнения лабораторной работы № 9, перейдём в него и со- здадим файл lab09-1.asm: (рис. ??).



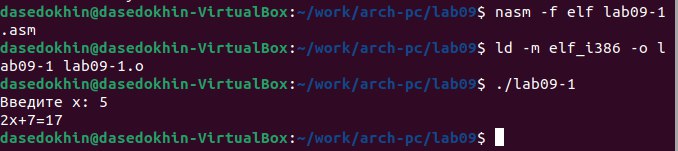
Создание каталога для лабораторной работы №9 и файла lab9-1.asm

1. В качестве примера рассмотрим программу вычисления арифметического выражения f(x) = 2x + 7 с помощью подпрограммы \_calcul. В данном примере x вводится с клавиатуры, а само выражение вычисляется в подпрограмме. Введём в файл lab9-1.asm текст программы из листинга 9.1. (рис. ??).



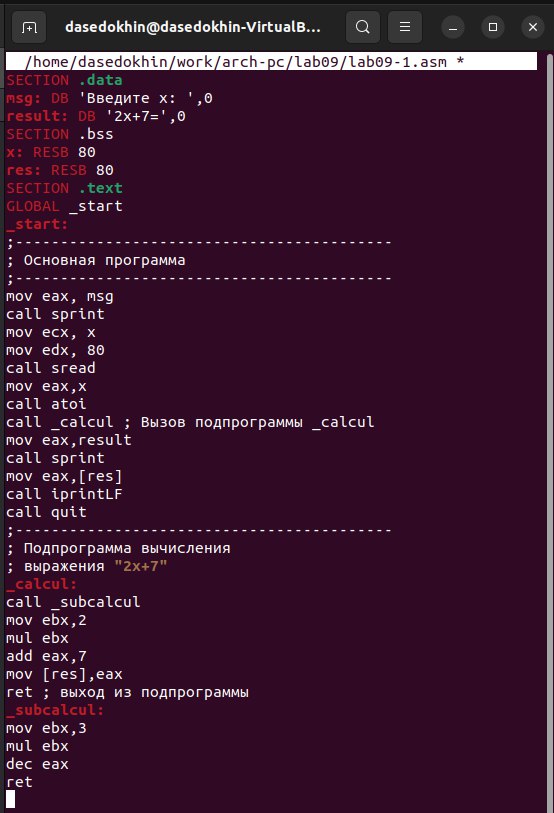
Ввод в файл lab9-1.asm текст программы из листинга 9.1

Создадим исполняемый файл и проверим его работу. (рис. ??).



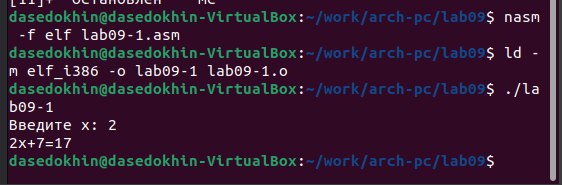
Создание и проверка исполняемого файла lab9-1.asm

Изменим 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 нее 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. Отредактируем текст программы (рис. ??).



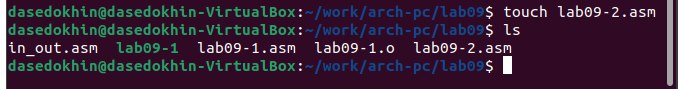
Изменение текста программы, добавив команду подпрограммы \_subcalcul

Создадим исполняемый файл и проверим его работу. (рис. ??).



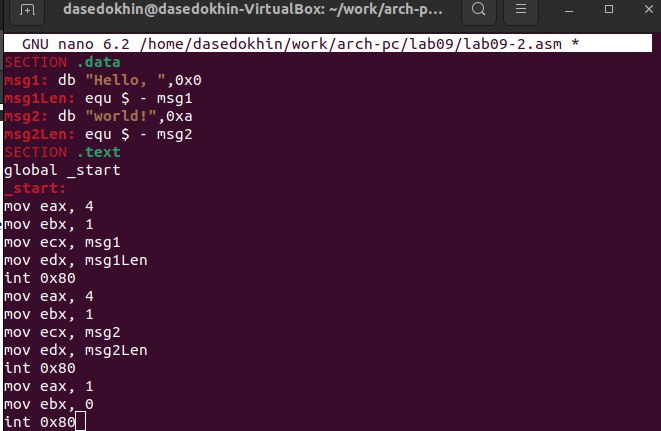
Создание и проверка исполняемого файла

Создадим файл lab09-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!) (рис. ??).



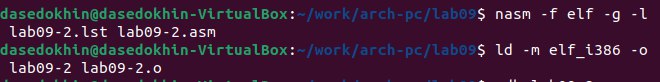
Создание файла lab09-2.asm

Введём в файл lab09-2.asm текст программы из листинга 9.2. (рис. ??).



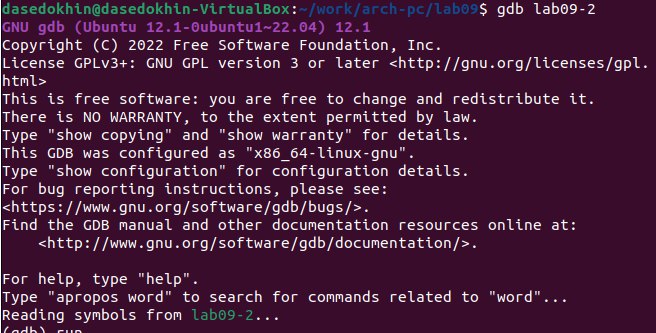
Ввод текст программы из листинга 9.2.

Получм исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’. (рис. ??).



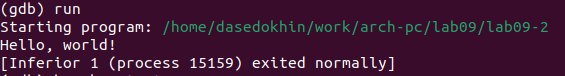
Получение исполняемого файла

Загрузим исполняемый файл в отладчик gdb (рис. ??).



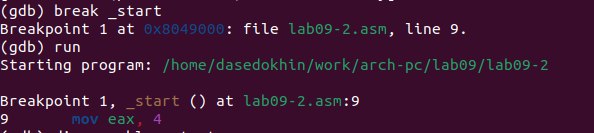
Загрузка файла в отладчик gdb

Проверим работу программы, запустив ее в оболочке GDB с помощью команды run (со- кращённо r): (рис. ??).



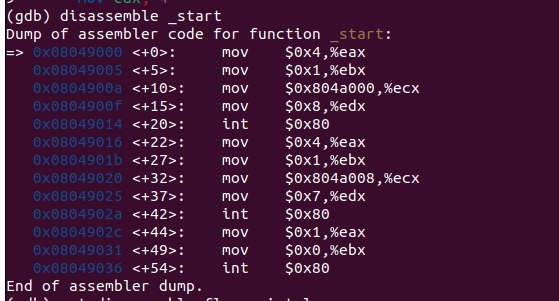
Проверка работы программы

Для более подробного анализа программы установим брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запустим её. (рис. ??).



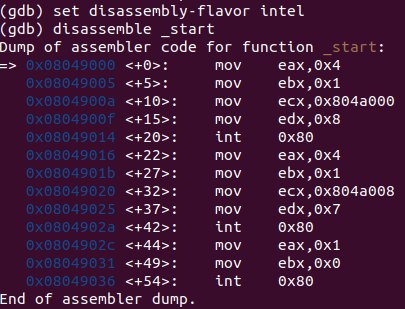
Подробный анализ программы \_start

Посмотрим дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. ??).



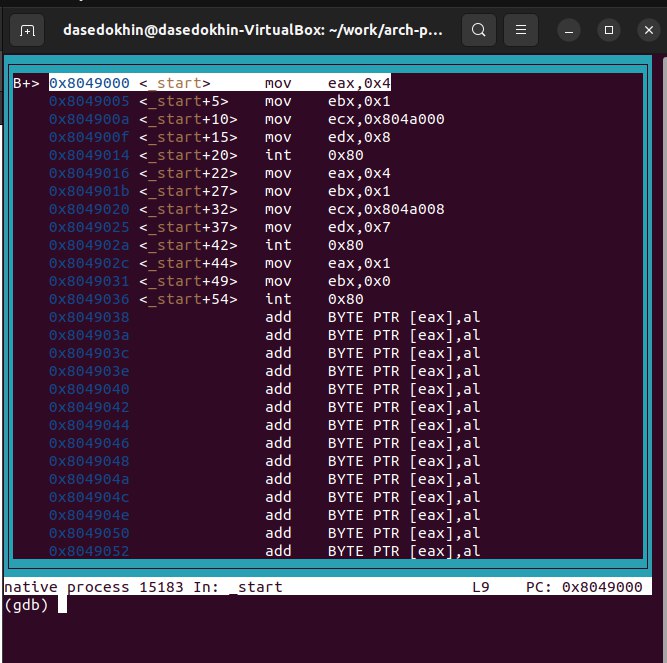
Дисассимилированный код программы

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

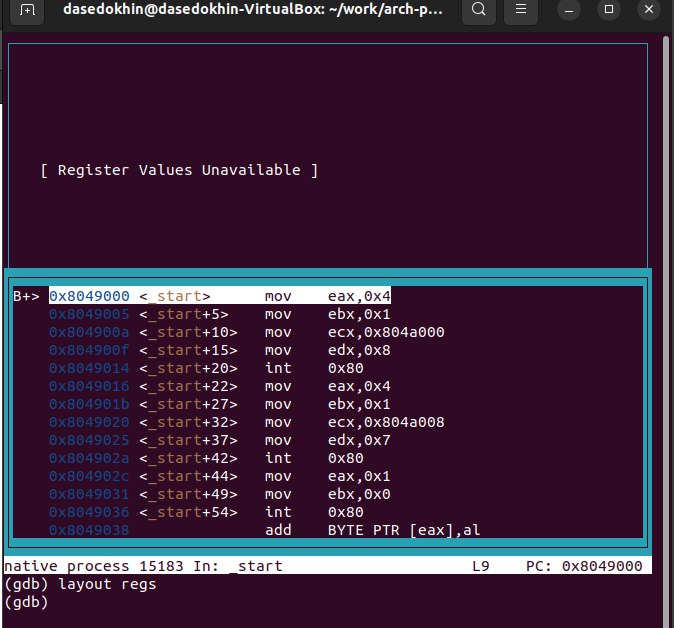


Переключение на отображение команд intel синтаксисом

В ATT имена регистров начинаются с символа %, а имена опрерандов с $, в то время как в intel используется привычный нам синтаксис. Включим режим псевдографики для более удобного анализа программы и выведем значения регисторов (рис. ?? ??).

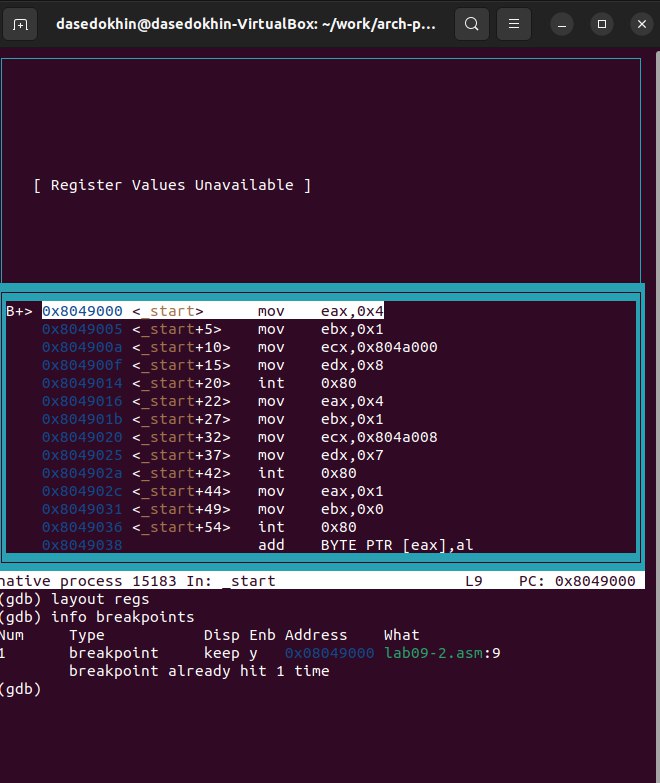


Режим пвсевдографики



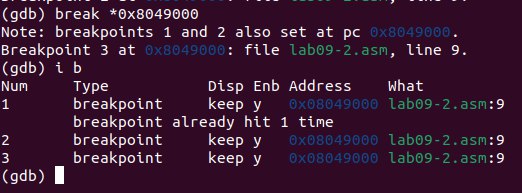
layout regs

Установить точку останова можно командой break (кратко b). Типичный аргумент этой команды — место установки. Его можно задать или как номер строки программы (имеет смысл, если есть исходный файл, а программа компилировалась с информацией об отладке), или как имя метки, или как адрес. Чтобы не было путаницы с номерами, перед адресом ставится «звёздочка»: На предыдущих шагах была установлена точка останова по имени метки (\_start). Про- верим это с помощью команды info breakpoints (кратко i b) (рис. ??).



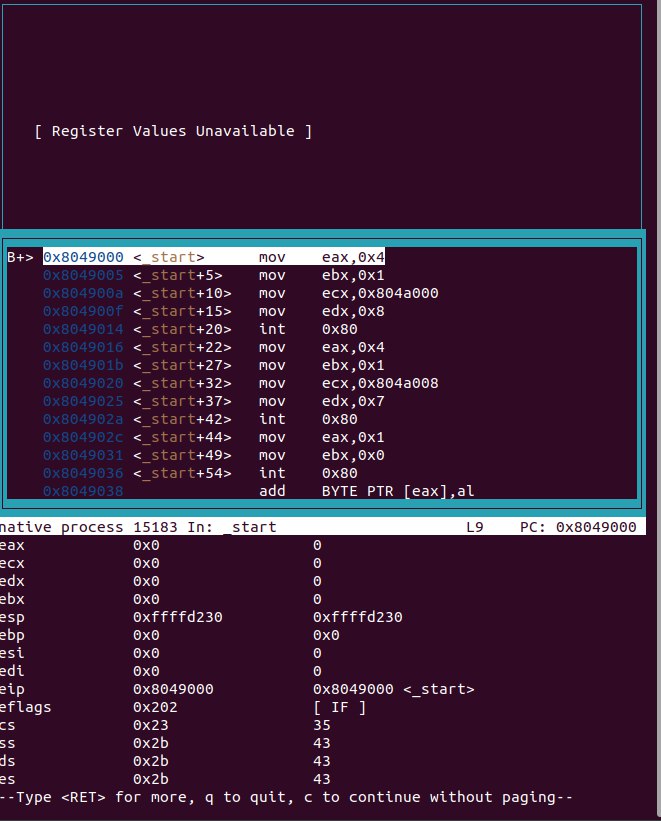
Проверка установки точки \_start

Установим еще одну точку останова по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции Определим адрес предпоследней инструкции (mov ebx,0x0) и установим точку останова. (рис. ??).



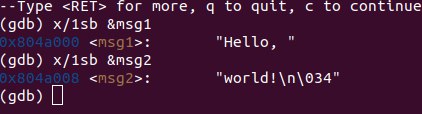
Определение адреса

Посмотрим информацию о всех установленных точках останова (рис. ??).



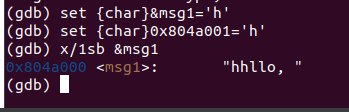
Просмотр информации о всех установленных точках останова

С помощью команды x & также можно посмотреть содержимое пере- менной. Посмотрим значение переменной msg1 и msg2 по имени (рис. ??).



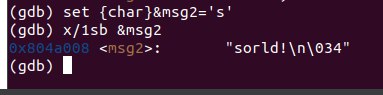
Просмотр значения msg1 и msg2

Изменить значение для регистра или ячейки памяти можно с помощью команды set, задав ей в качестве аргумента имя регистра или адрес. При этом перед именем регистра ставится префикс $, а перед адресом нужно указать в фигурных скобках тип данных (раз- мер сохраняемого значения; в качестве типа данных можно использовать типы языка Си). Изменим первый символ переменной msg1 (рис. ??).



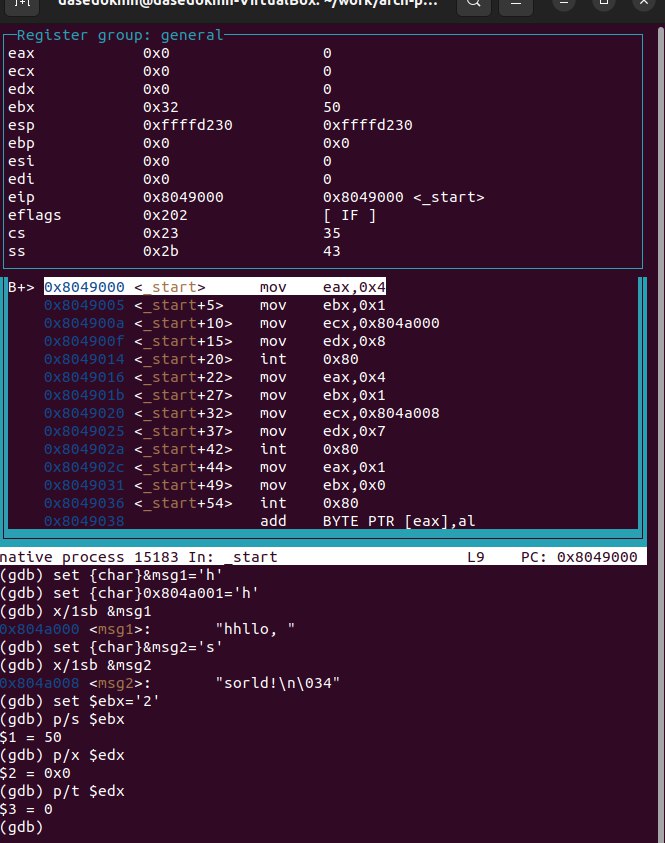
Изменение первого символа переменной msg1

Заменим любой символ во второй переменной msg2. (рис. ??).



Изменение первого символа переменной msg2

С помощью команды set изменим значение регистра ebx (рис. ??).



Изменение значение регистра ebx

Разница вывода команд p/s $ebx отличается тем, что в первом случае мы переводим символ в его строковой вид, а во втором случае число в строковом виде не изменяется.

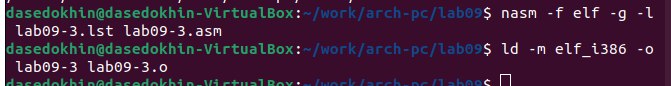
Завершим выполнение программы с помощью команды continue (сокращенно c) или stepi (сокращенно si) и выйдем из GDB с помощью команды quit (сокращенно q).

Скопируем файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm (рис. ??).

Копирование файла lab8-2.asm в файл lab09-3.asm

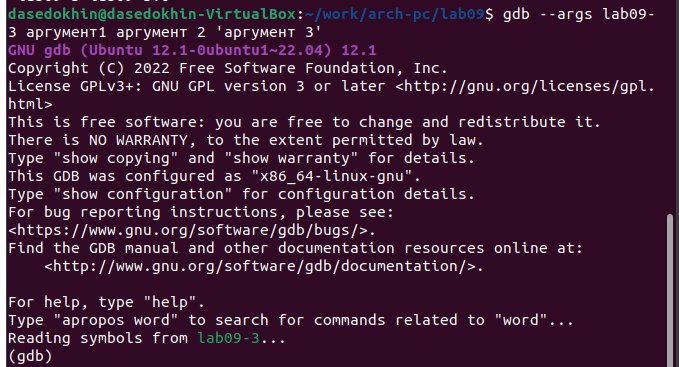
Копирование файла lab8-2.asm в файл lab09-3.asm

Создадим исполняемый файл (рис. ??).



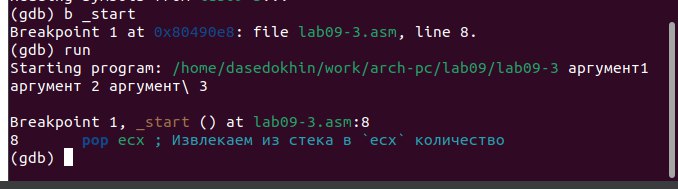
Создание исполняемого файла

Для загрузки в gdb программы с аргументами необходимо использовать ключ –args. Загрузим исполняемый файл в отладчик, указав аргументы (рис. ??).



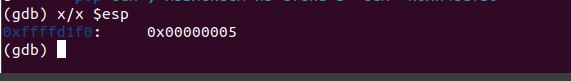
Загрузка исполяемого файла в отладчик

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 ее. (рис. ??).



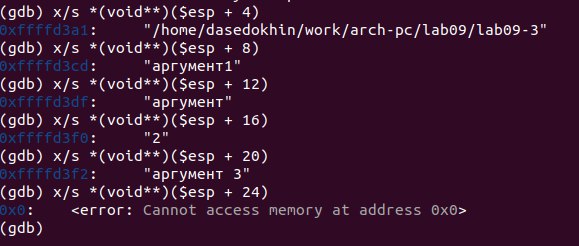
Установка точки останова в программе и её запуск

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



Адрес вершины стека

Как видно, число аргументов равно 5 – это имя программы lab09-3 и непосредственно аргументы: аргумент1, аргумент, 2 и ‘аргумент 3’. Посмотрим остальные позиции стека – по адесу [esp+4] располагается адрес в памяти где находиться имя программы, по адесу [esp+8] храниться адрес первого аргумента, по аресу [esp+12] – второго и т.д. (рис. ??).

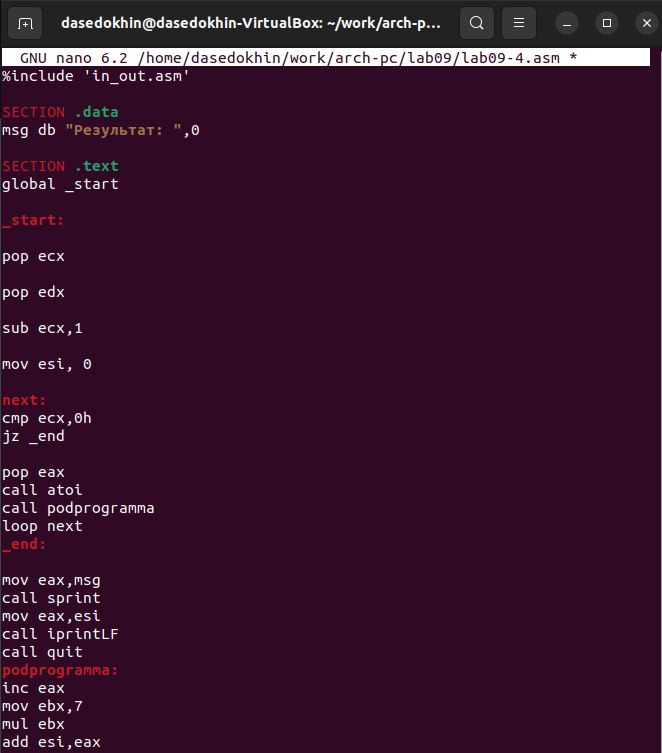


Просмотр позиций стека

# 3 Задание для самостоятельной работы

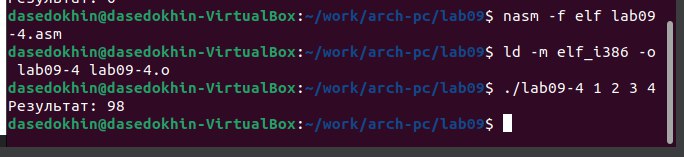
Преобразуем программу из лабораторной работы №8 (Задание №1 для самостоятель- ной работы), реализовав вычисление значения функции f(x) как подпрограмму.

Для этого, сначала создадим файл lab09-4.asm и за основу текста его программы возьмём текст программы из прошлой лабораторной работы файла srab1.asm и отредактируем текст под нашу задачу (рис. ??).



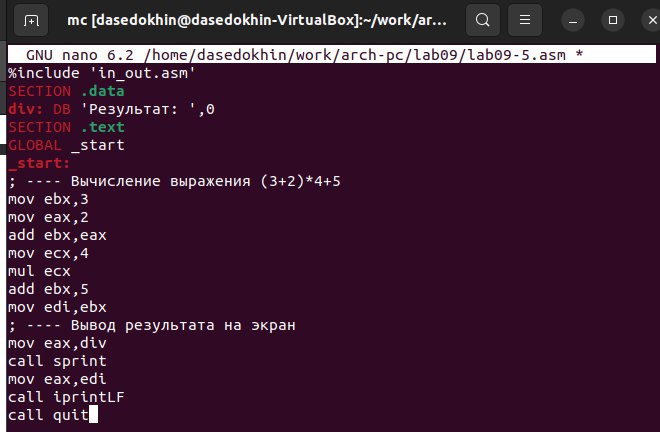
Текст файла lab09-4.asm

Создадим исполняемый файл и проверим его работу (рис. ??).



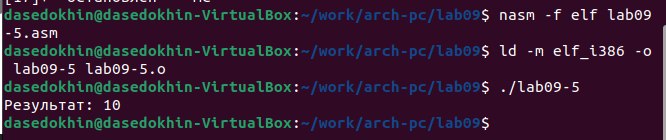
Создание исполняемого файла и его проверка

Создадим файл lab09-5.asm и введём в него екст программы из листинга 9.3 (рис. ??).



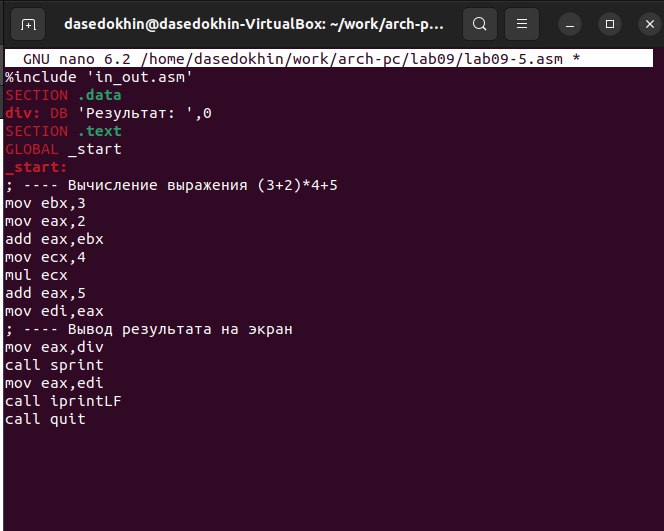
Текст файла lab09-5.asm

В листинге 9.3 приведена программа вычисления выражения (3 + 2) ∗ 4 + 5. При запуске данная программа дает неверный результат. Проверим, создав исполняемый файл. (рис. ??).



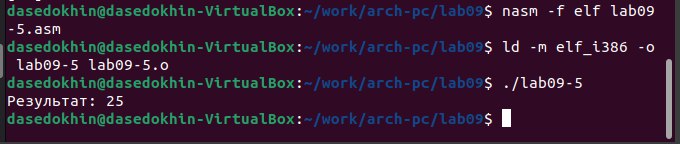
Создание и проверка исполяемого файла

Получим исполняемый файл через gpd. Проверим с помощью команды continue каждую точку останова и значения регистров. На основе этого внесём изменения в программу (рис. ??).



Правильный код программы lab09-5.asm

Создадим исполняемый файл и проверим его работу (рис. ??).



Создание исполняемого файла и его проверка

# 4 Выводы

Я приобрёл навыки написания программ с использованием подпрограмм и ознакомился с методами отладки при помощи GDB и его основными возможностями.