Лабораторная работа №10.

Понятие подпрограммы. Отладчик GDB.

Боровиков Даниил Александрович

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

Создадим каталог для программ лабораторной работы № 10, перейдем в него и создадим файл lab10-1.asm(рис. 1)

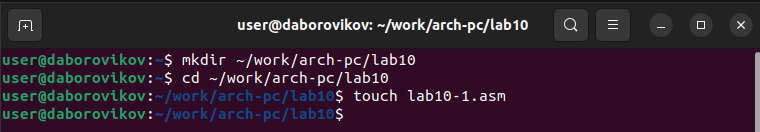


Рис. 1: Создание файла lab10-1.asm в соответствующем каталоге

Введем в файл lab10-1.asm текст программы из листинга 10.1.(рис. 2)

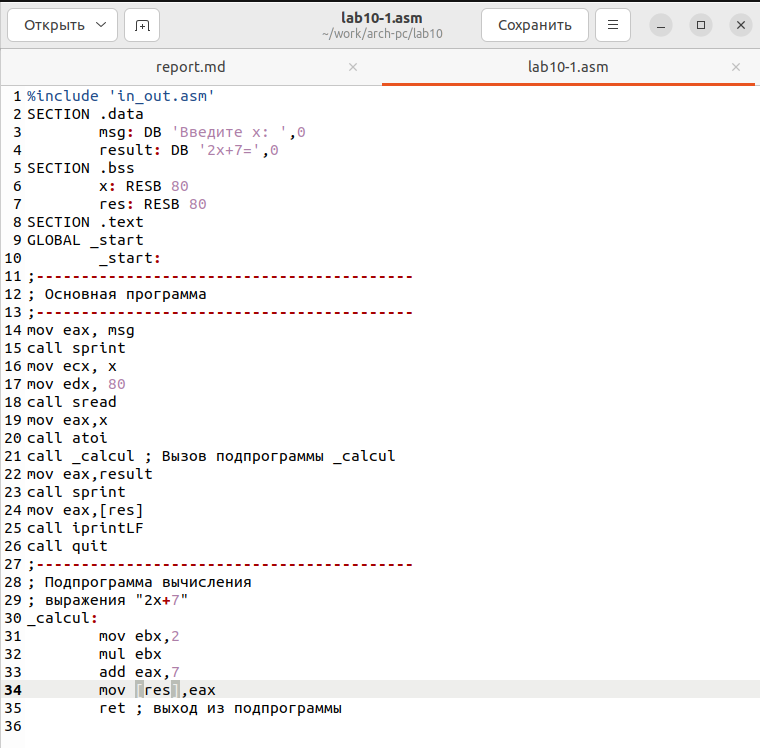


Рис. 2: Текст программы из листинга 10.1.

Создадим исполняемый файл и запустим его.(рис. 3)

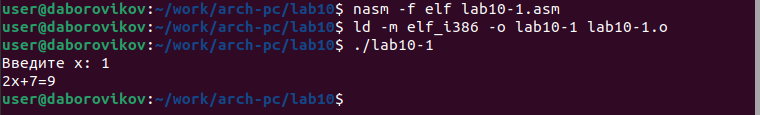


Рис. 3: Запуск исполняемого файла lab10-1.asm

Далее 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.(рис. 4)

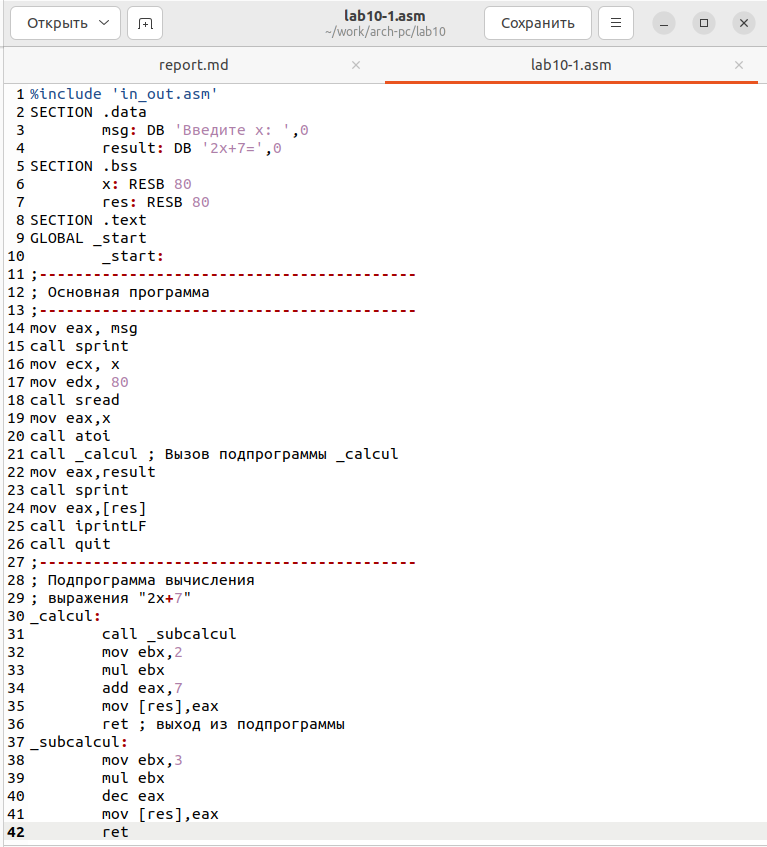


Рис. 4: Текст измененной программы

Создадим исполняемый файл исправленного текста программы lab10-1.asm и запустиv его.(рис. 5)

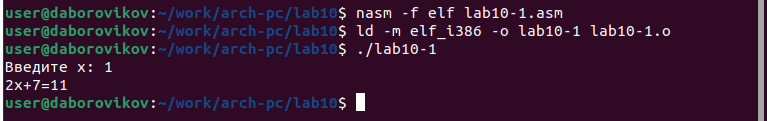


Рис. 5: Запуск исправленного исполняемого файла lab10-1.asm

Создадим файл lab10-2.asm с текстом программы из Листинга 10.2. (Програм- ма печати сообщения Hello world!):(рис. 6)

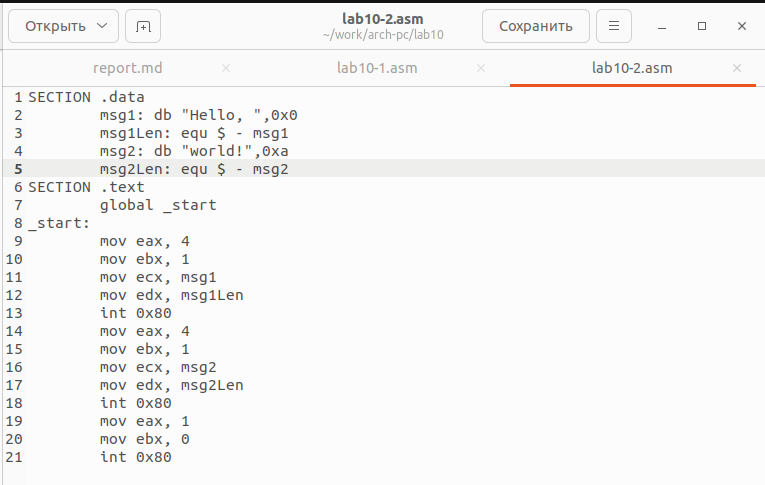


Рис. 6: Листинг программы 10.2

Получим исполняемый файл. Для работы с GDB в исполняемый файл добавим отладочную информацию, для этого трансляцию программ проведем с ключом ‘-g’.

nasm -f elf -g -l lab10-2.lst lab10-2.asm

ld -m elf\_i386 -o lab10-2 lab10-2.o

Загрузим исполняемый файл в отладчик gdb:

user@dk4n31:~$ gdb lab10-2

Проверим работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r):

Для более подробного анализа программы установим брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запустим её.(рис. 7)

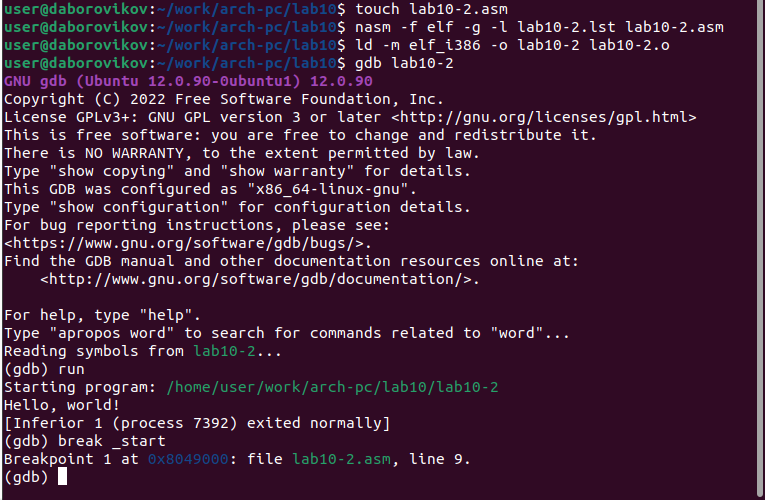


Рис. 7: Запуск измененного исполняемого файла lab10-2.asm

Посмотрим дисассимилированный код программы с помощью команды disassemble начиная с метки \_start

(gdb) disassemble \_start

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

(gdb) set disassembly-flavor intel

(gdb) disassemble \_start(рис. 8)

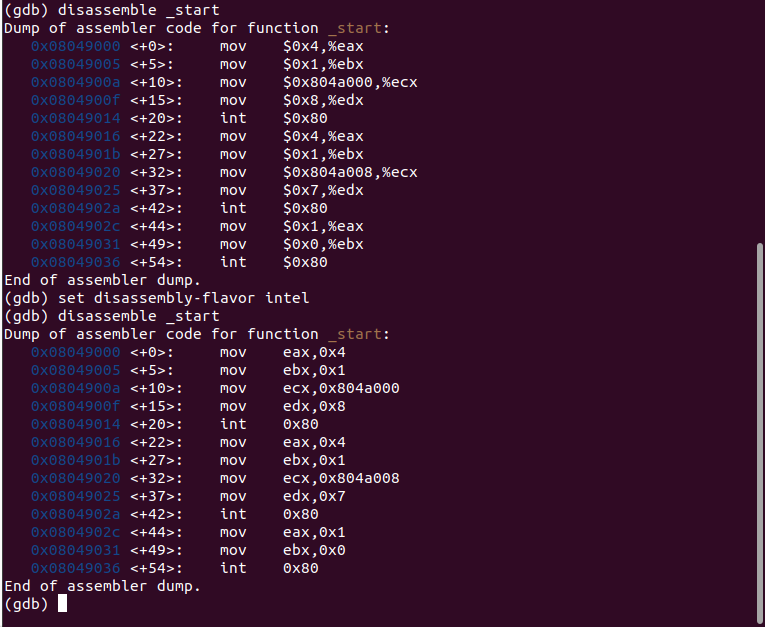


Рис. 8: Дисассимилированный код программы

Синтаксисы машинных команд в режимах ATT и Intel заключаются в наличии символов “$” и “&” в ATT режиме.

Включим режим псевдографики для более удобного анализа программы(рис. 9)

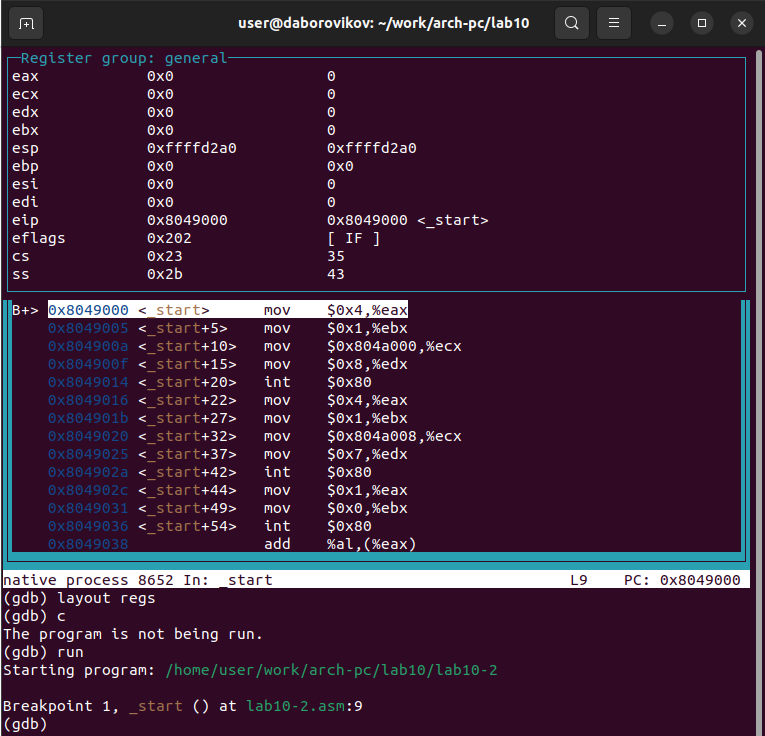


Рис. 9: Режим псевдографики

Проверим брейкпоинты командой info breakpoints(рис. 10)

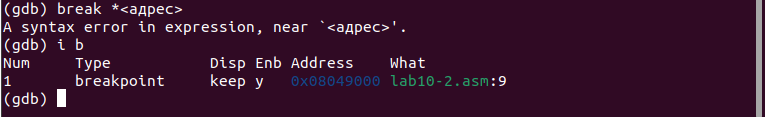


Рис. 10: Брейкпоинты

Установим еще один брейкпоинт по адресу инструкции и проверим(рис. 11)

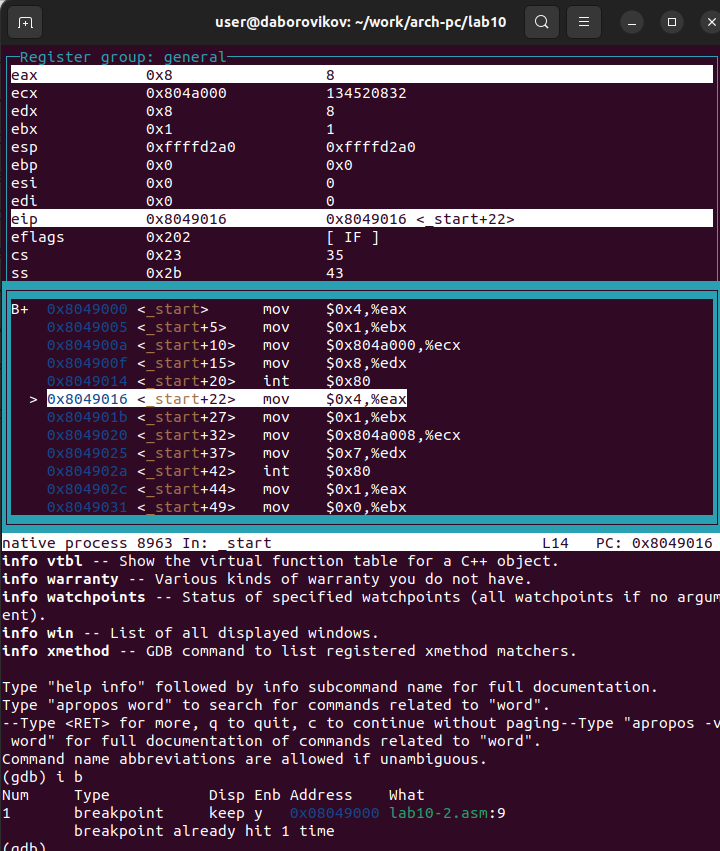


Рис. 11: Дополнительный брейкпоинт

Выполним 5 инструкций с помощью команды stepi (или si) и проследите за изменением значений регистров. Посмотрим содержимое регистров с помощью команды inforegisters (рис. 12)

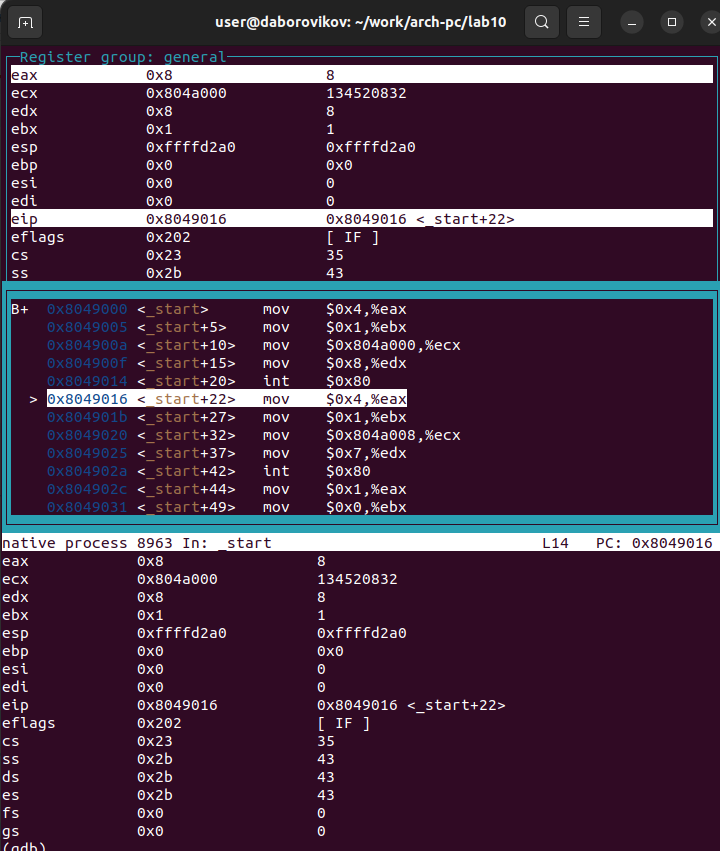


Рис. 12: Содержимое регистров

Посмотрим значение переменной msg1 по имени, а msg2 по адресу (рис. 13)

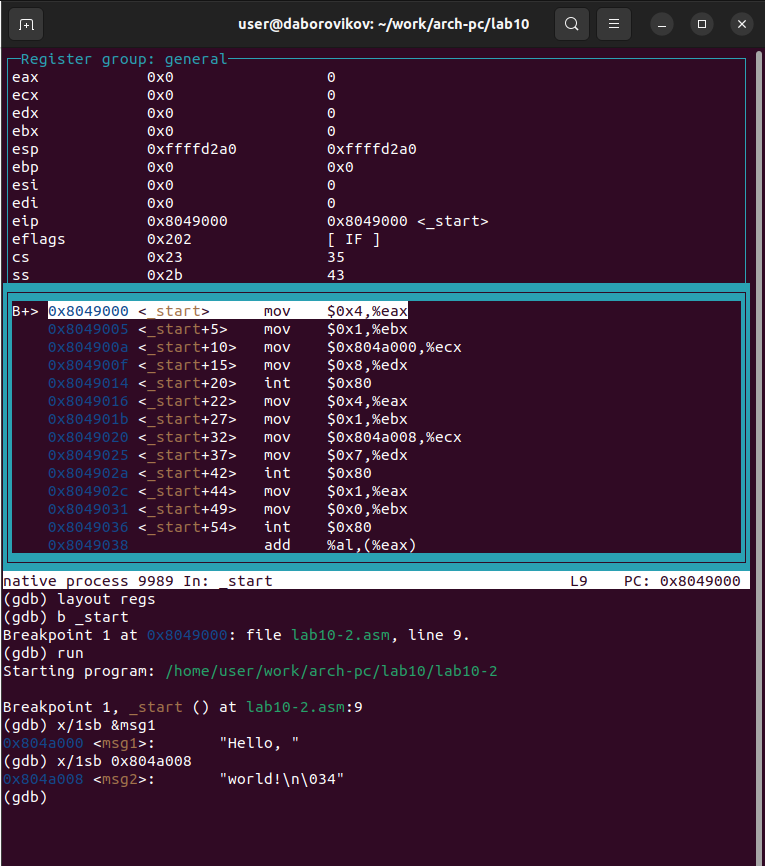


Рис. 13: Значения переменных

Изменим первый символ переменной msg1(рис. 14)

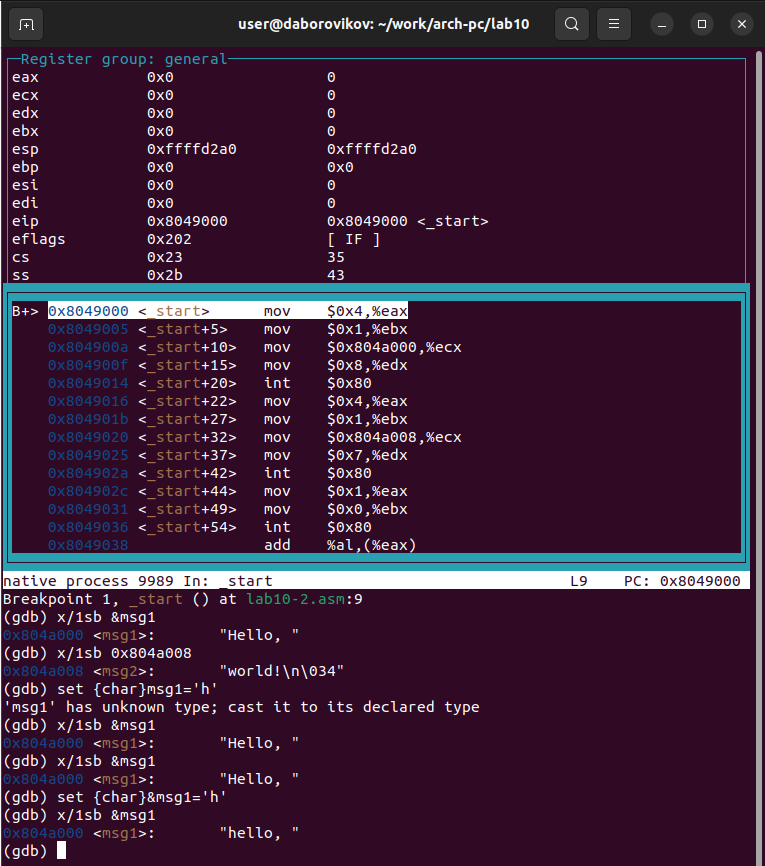


Рис. 14: Замена первого символа msg1

Изменим символы переменной msg2(рис. 15)

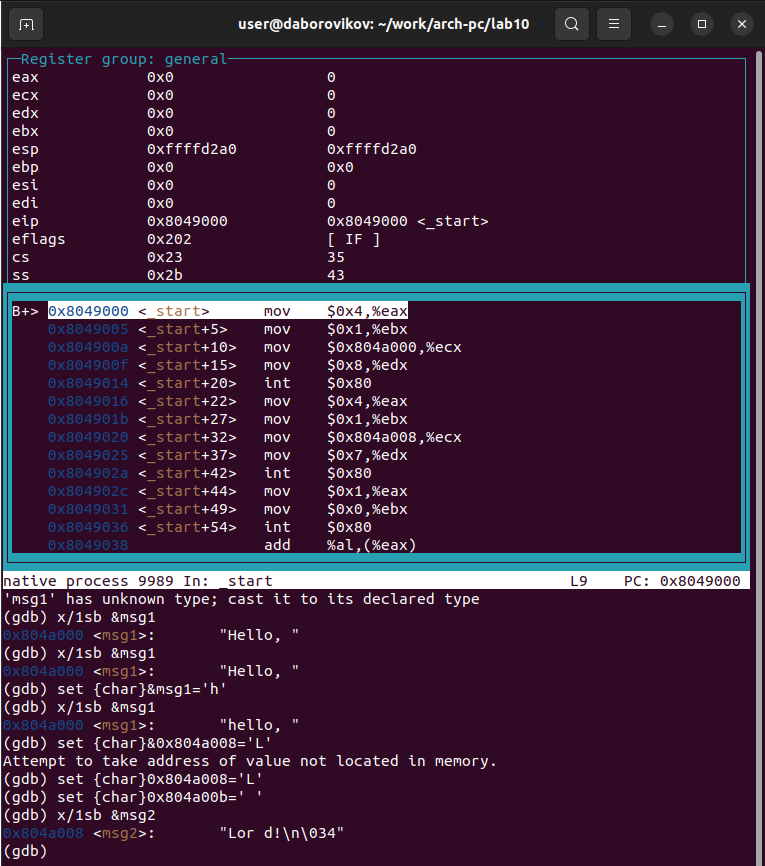


Рис. 15: Замена символов msg2

Выведеv в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx.(рис. 16)



Рис. 16: edx в различных форматах

С помощью команды set изменим значение регистра ebx:(рис. **¿fig:017?**)

 Разница в командах в том, что примваиваем значение числа во втором случае, а в первом символ ‘2’.

Завершим выполнение программы с помощью команды continue (сокра- щенно c) или stepi (сокращенно si) и выйдите из GDB с помощью команды quit (сокращенно q)

Скопируем файл lab9-2.asm, созданный при выполнении лабораторной ра- боты №9, с программой выводящей на экран аргументы командной строки (Листинг 9.2) в файл с именем lab10-3.asm и создадим исполняемый файл.(рис. 17)

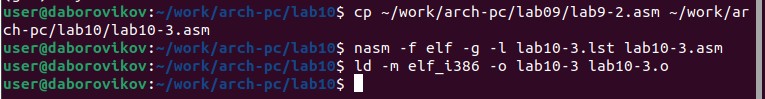


Рис. 17: Создание исполняемого файла

Для загрузки в gdb программы с аргументами используем ключ –args. Загрузим исполняемый файл в отладчик, указав аргументы(рис. 18)

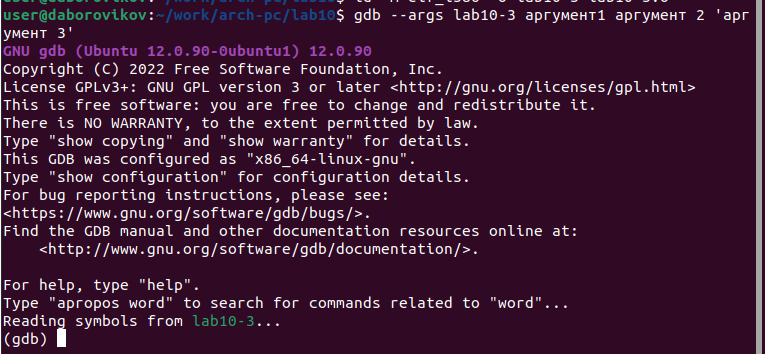


Рис. 18: Загрузка исполняемого файла в отладчик

Для начала установим брейкпоинт перед первой инструкцией в программе и запустим ее.(рис. 19)

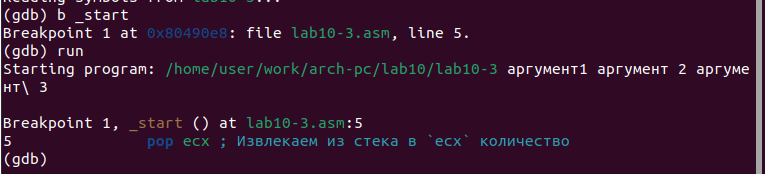


Рис. 19: Брейкпоинт

Посмотриv позиции стека c шагом +4(рис. 20)

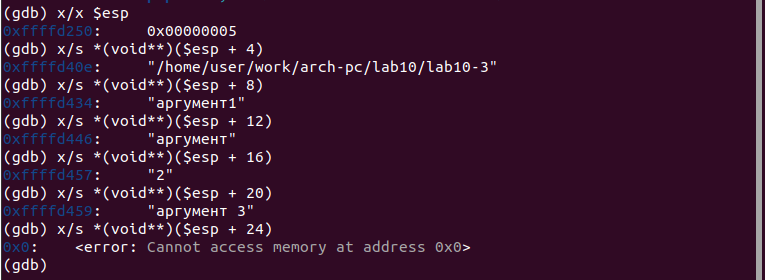
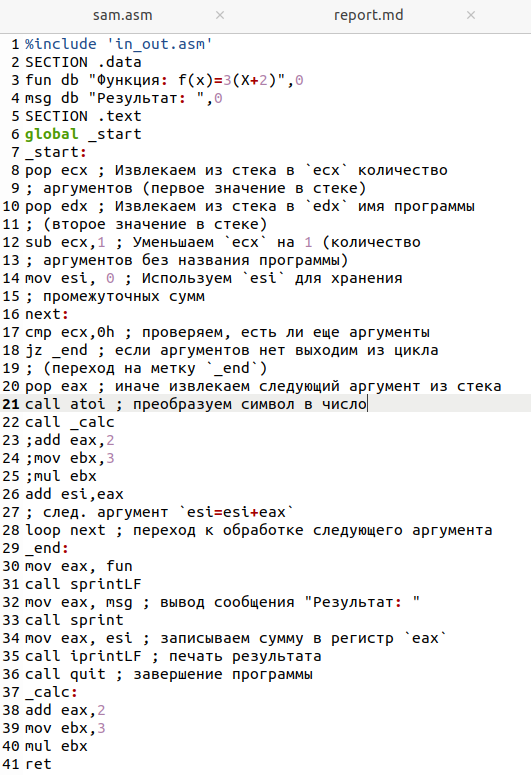


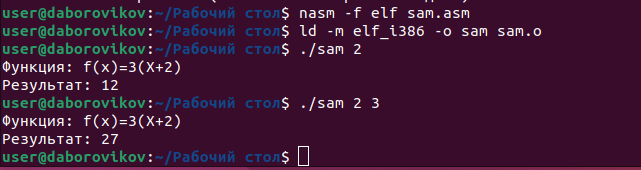
Рис. 20: Просмотр позиций стека

Шаг равен размеру переменной - 4 байтам. #Самостоятельная работа

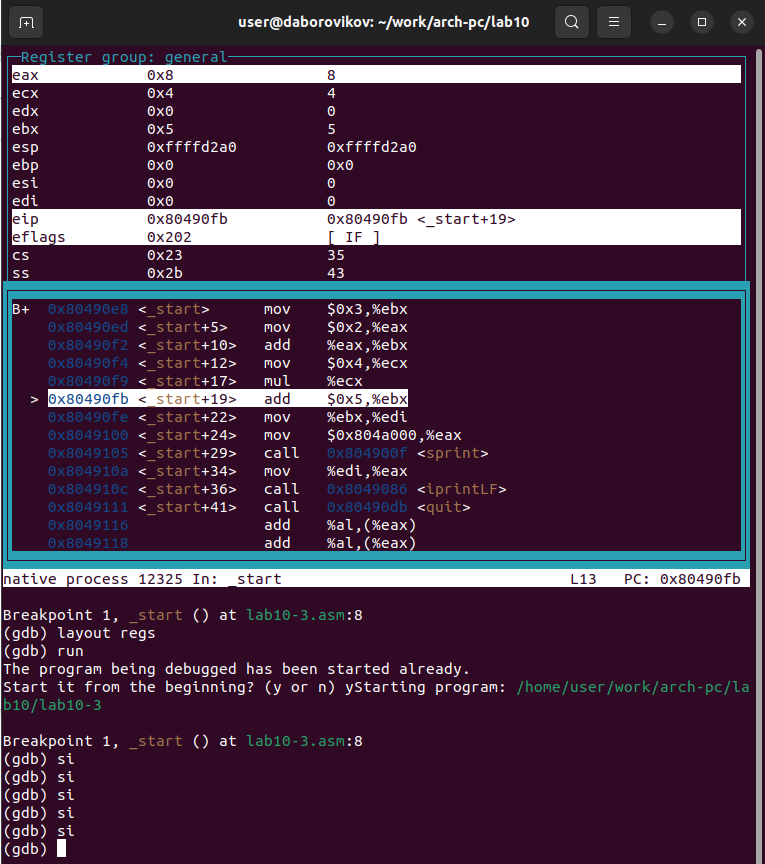
Преобразуем программу из лабораторной работы 9 задание 1 для сам работы, чтобы вычисления были в подпрограмме(рис. **¿fig:022?**)



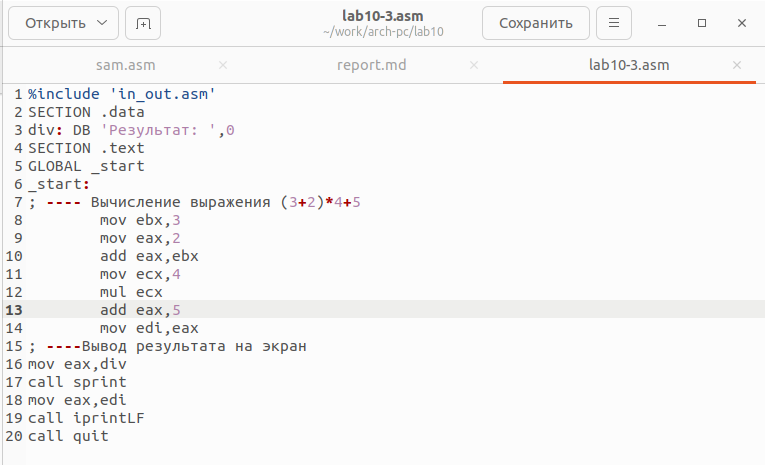
Создадим испоняемый файл для проверки программы(рис. **¿fig:030?**)



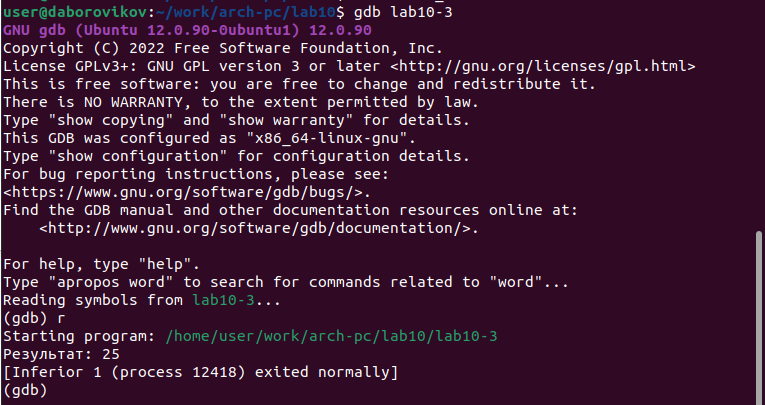
Создадим файл для задания 2 сам. работы. И ввдём программу из листинга 10.3(рис. **¿fig:023?**)



Проверим программу с помощью отладчика и найдем ошибку(рис. **¿fig:024?**)



Исправим ошибку(рис. **¿fig:025?**)



Запустим рабочую программу(рис. **¿fig:026?**)

# 3 Выводы

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

В ходе лабораторной работы мы приобрели навыки написания программ с использованием подпрограмм, познакомились с методами отладки при помощи GDB и его основными возможностями.

https://github.com/daBorovikov/study\_2022-2023\_arh-pc-