Отчёт по лабораторной работе 9

Архитектура компьютеров

Исмаил М. А. НКАбд-03-24

Содержание

# 1 Цель работы

Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

# 2 Выполнение лабораторной работы

## 2.1 Реализация подпрограмм в NASM

Для выполнения лабораторной работы №9 я создал новую папку и перешел в нее. Затем я создал файл с именем lab9-1.asm.

В качестве примера была рассмотрена программа, которая вычисляет арифметическое выражение с использованием подпрограммы calcul. Значение переменной вводится с клавиатуры, а вычисление производится внутри подпрограммы. (рис. 1) (рис. 2)

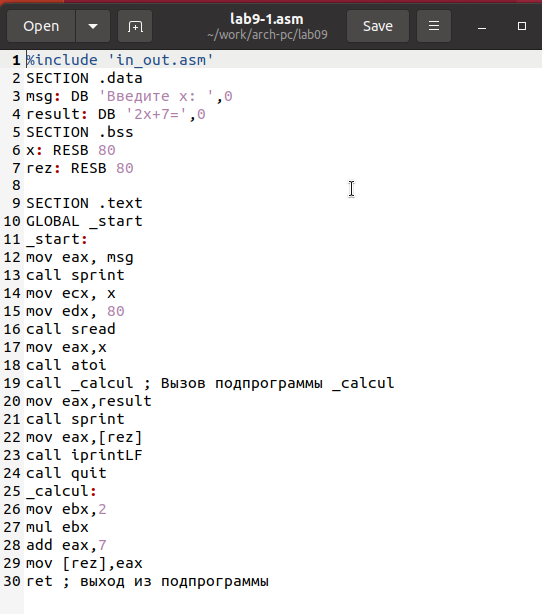


Рис. 1: Текст программы lab9-1.asm

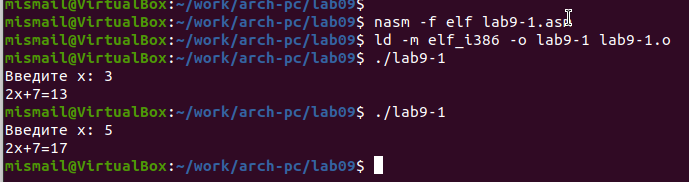


Рис. 2: Запуск программы lab9-1.asm

Далее я модифицировал программу, добавив подпрограмму subcalcul внутрь подпрограммы calcul. Это позволило вычислять составное выражение , где , а . Значение вводится с клавиатуры. (рис. 3) (рис. 4)

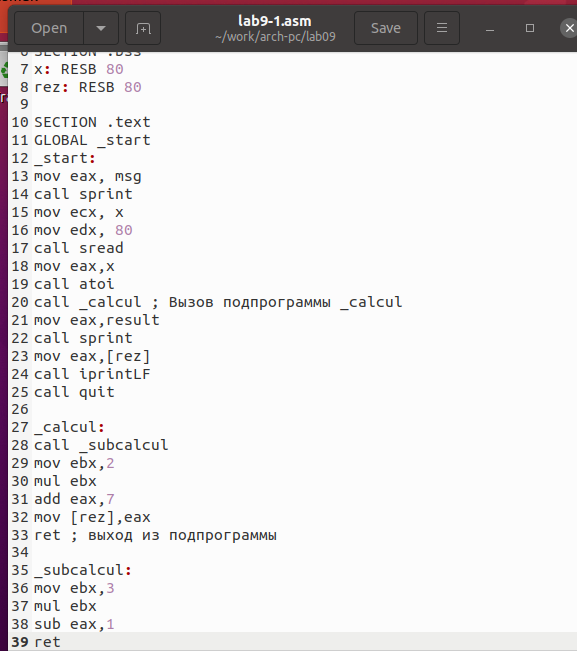


Рис. 3: Модифицированная программа lab9-1.asm

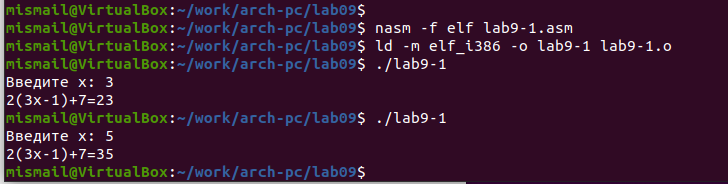


Рис. 4: Запуск модифицированной программы lab9-1.asm

## 2.2 Отладка программы с помощью GDB

Я создал файл lab9-2.asm, в котором содержится программа из Листинга 9.2. Она отвечает за вывод сообщения “Hello world!” на экран. (рис. 5)

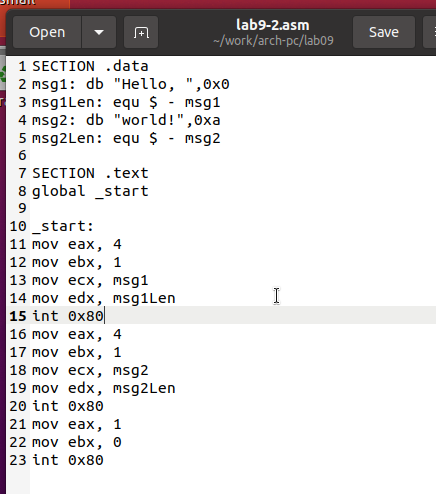


Рис. 5: Код программы lab9-2.asm

После компиляции с ключом -g для добавления отладочной информации я загрузил исполняемый файл в GDB. Запустил программу с помощью команды run или r. (рис. 6)

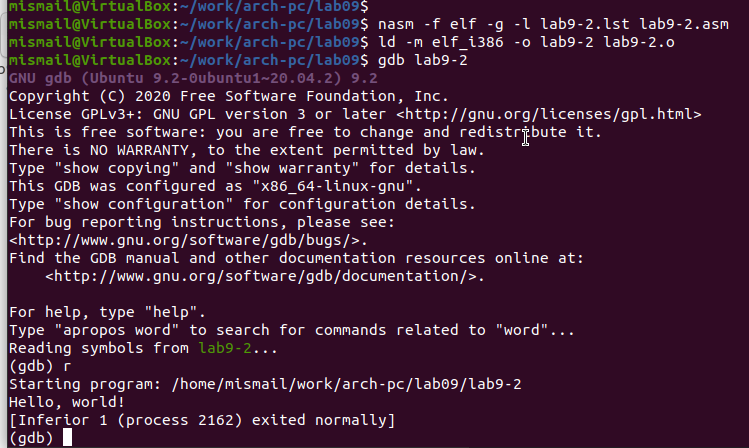


Рис. 6: Запуск программы lab9-2.asm в GDB

Для анализа программы я установил точку остановки на метке \_start и запустил выполнение. Затем изучил дизассемблированный код программы. (рис. 7) (рис. 8)

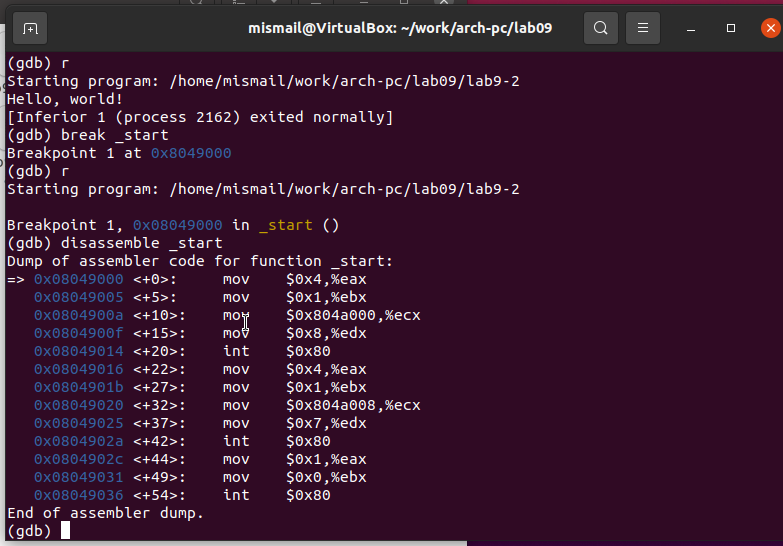


Рис. 7: Дизассемблированный код программы

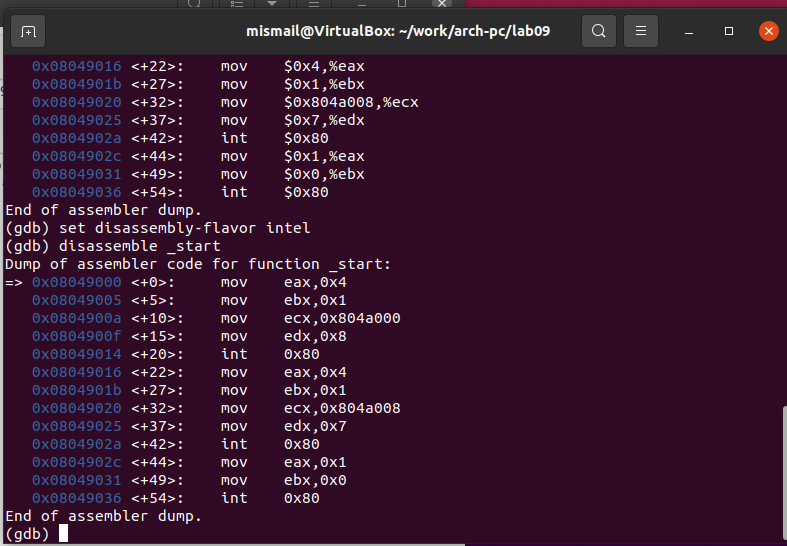


Рис. 8: Дизассемблированный код в Intel-синтаксисе

Для проверки точки останова я использовал команду info breakpoints (i b). Установил дополнительную точку останова по адресу инструкции mov ebx, 0x0. (рис. 9)

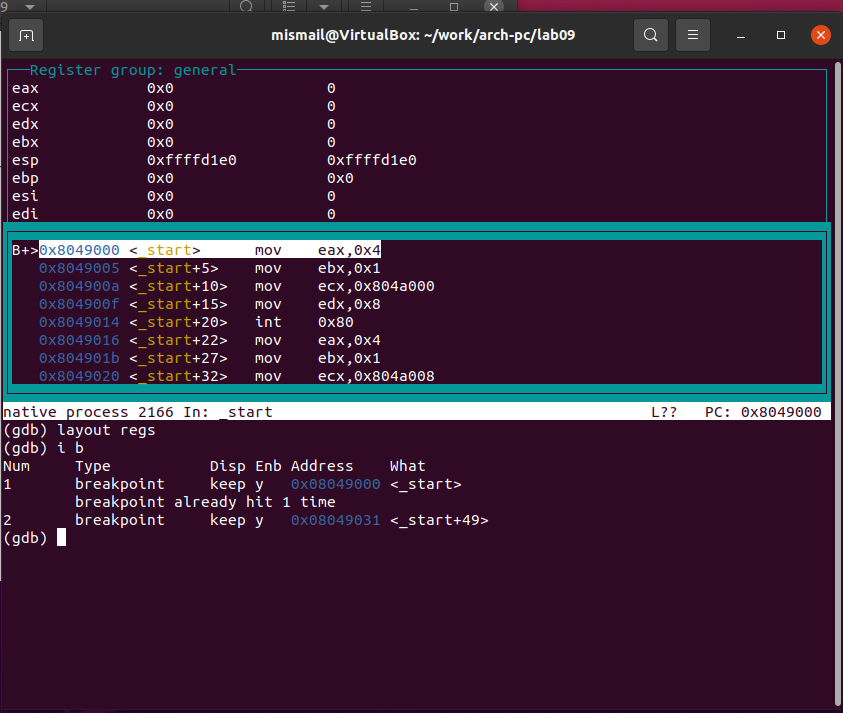


Рис. 9: Настройка точки останова

С помощью команды stepi (si) выполнил пошаговую отладку, отслеживая изменения регистров. (рис. 10) (рис. 11)

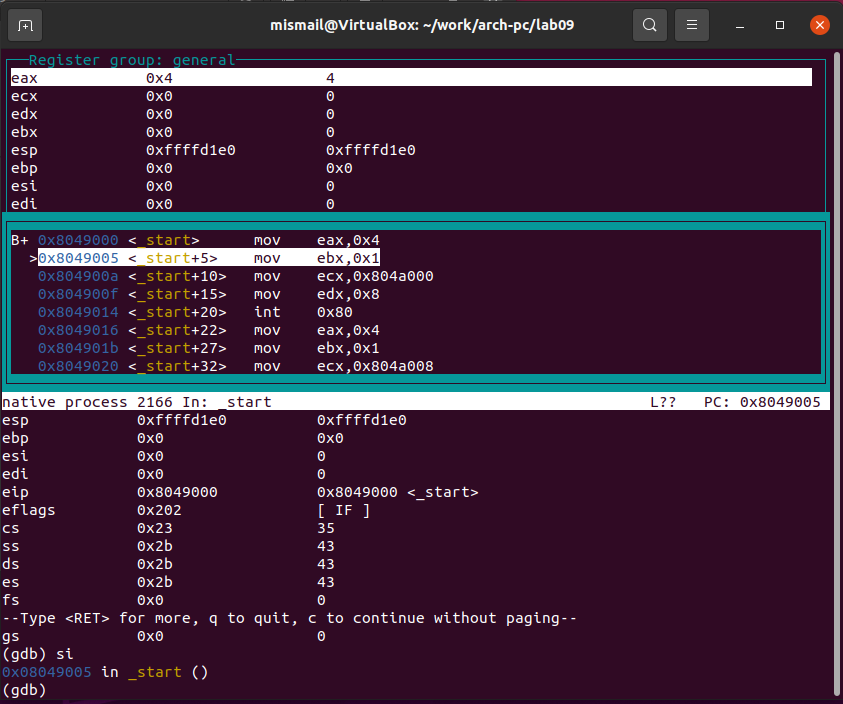


Рис. 10: Отслеживание изменений регистров

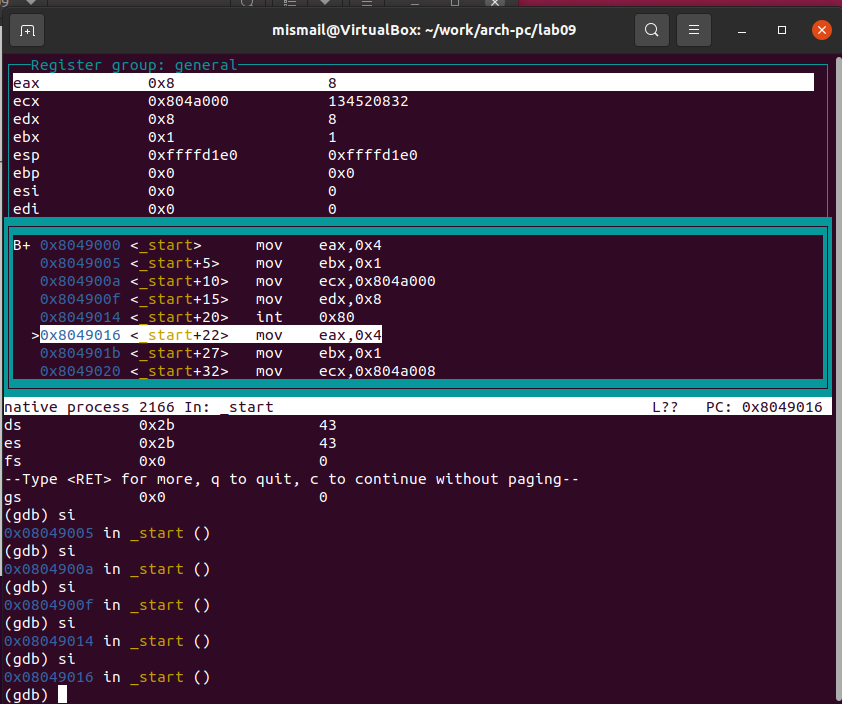


Рис. 11: Детальный анализ регистров

Я также просмотрел значение переменной msg1 по имени и изменил первый символ переменной с помощью команды set. (рис. 12) (рис. 13)

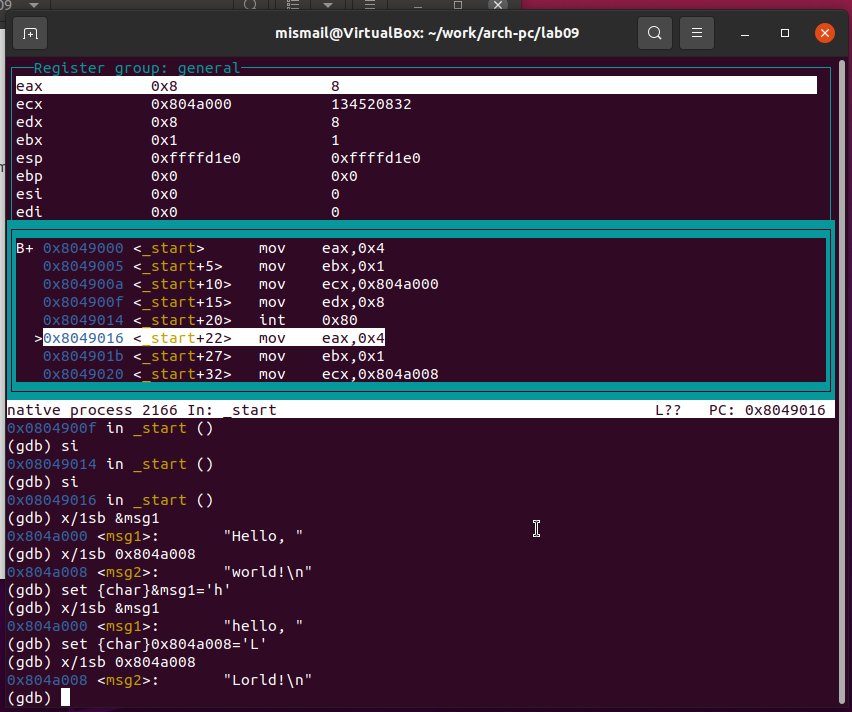


Рис. 12: Изменение значения переменной msg1

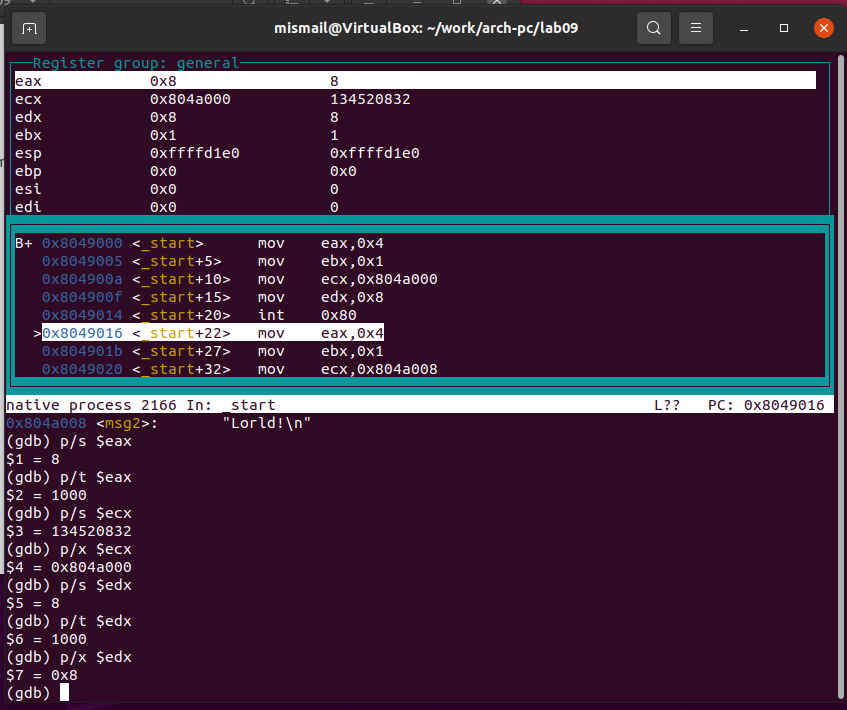


Рис. 13: Просмотр регистра после изменений

Для проверки программы с аргументами я скопировал файл lab8-2.asm из лабораторной работы №8, создал исполняемый файл и загрузил его в GDB с помощью ключа –args. Затем исследовал стек, где хранились адреса аргументов. (рис. 14)

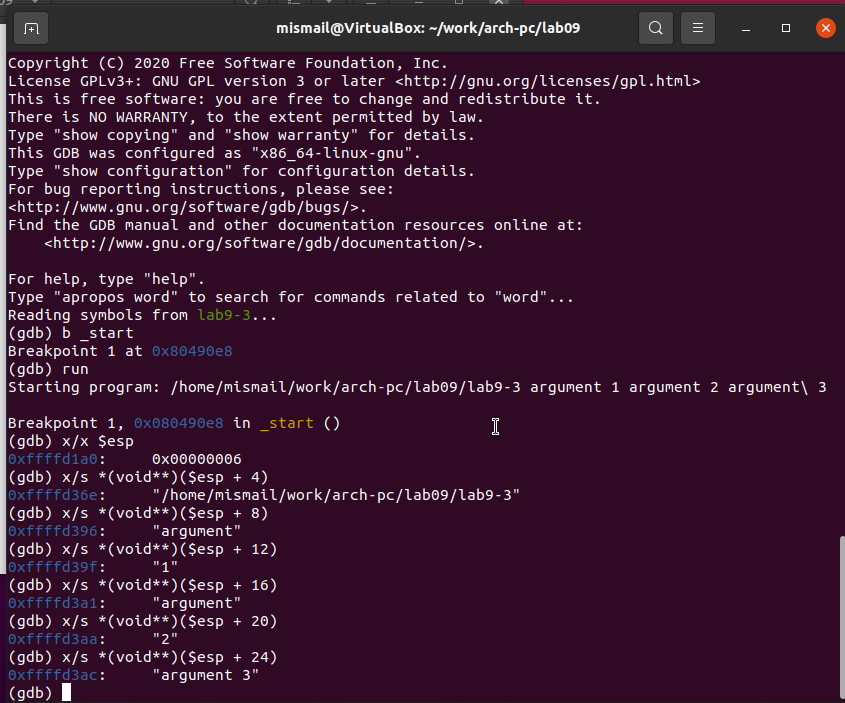


Рис. 14: Анализ стека программы

## 2.3 Задание для самостоятельной работы

Я модифицировал программу из лабораторной работы №8, добавив вычисление функции в виде подпрограммы. (рис. 15) (рис. 16)

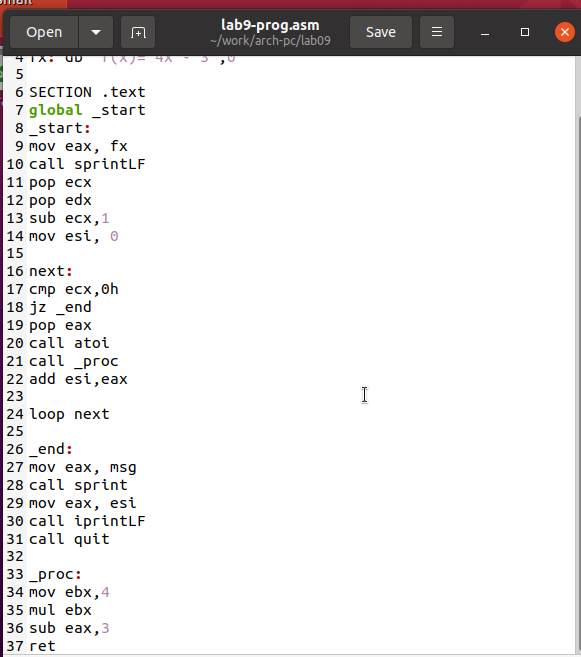


Рис. 15: Код программы lab9-prog.asm

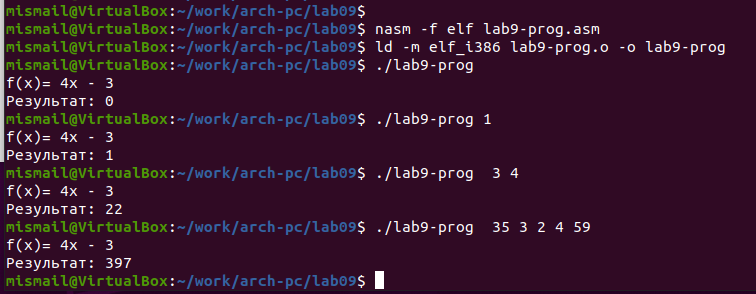


Рис. 16: Запуск программы lab9-prog.asm

При запуске программы я обнаружил ошибку: результат вычислений был неверным. Анализ с помощью GDB показал, что аргументы инструкции add перепутаны, а по окончании программы значение регистра ebx вместо eax отправляется в edi. (рис. 17) (рис. 18)

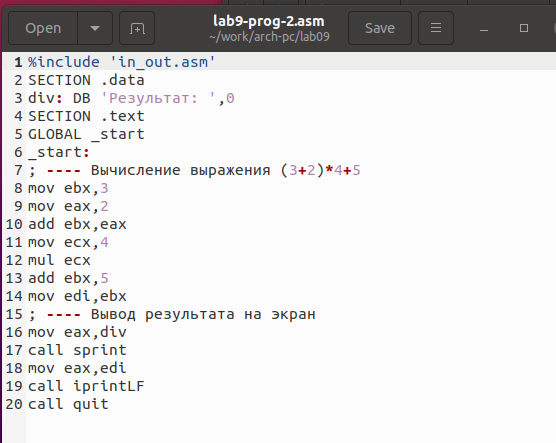


Рис. 17: Код с ошибкой

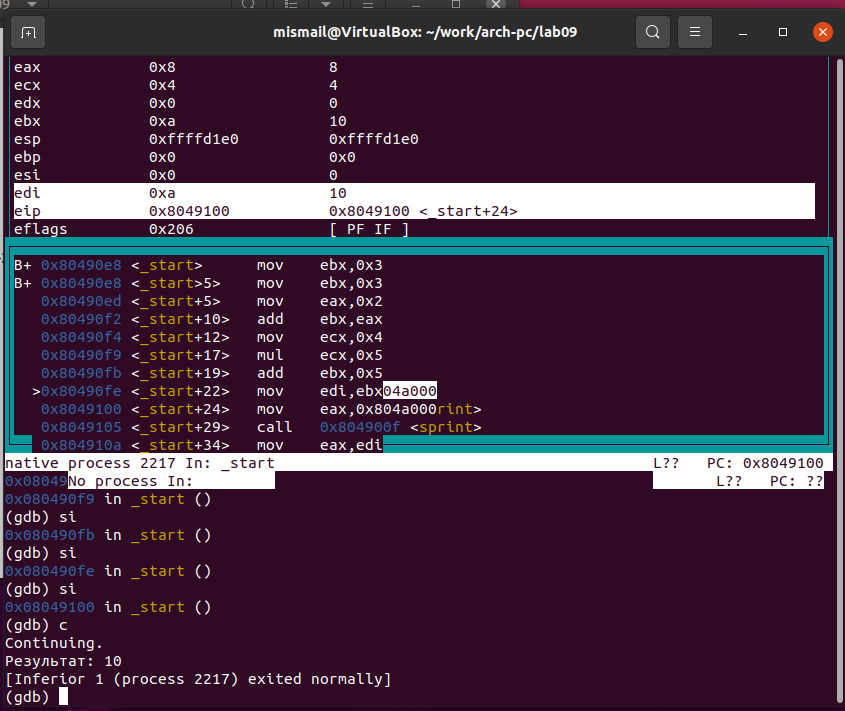


Рис. 18: Процесс отладки программы

После исправления ошибок я проверил работу программы. (рис. 19) (рис. 20)

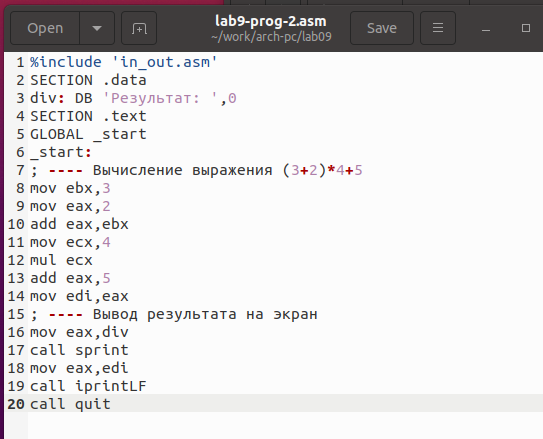


Рис. 19: Исправленный код программы

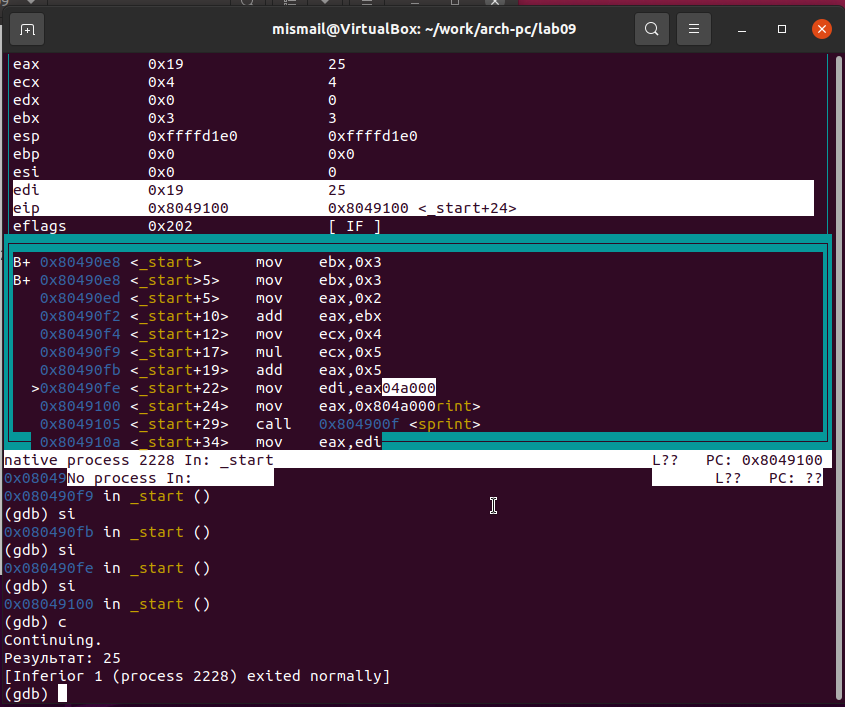


Рис. 20: Проверка исправленного кода

# 3 Выводы

Я освоил работу с подпрограммами и отладчиком GDB, научился находить и исправлять ошибки в коде с помощью анализа стеков, регистров и дизассемблированного кода.