Отчёт по лабораторной работе 9

Архитектура компьютеров

Хаммудех Салех

Содержание

Список иллюстраций

Список таблиц

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 $f(x) = 2x + 7$ с использованием подпрограммы calcul. Значение переменной x вводится с клавиатуры, а вычисление производится внутри подпрограммы. (рис. 2.1) (рис. 2.2)

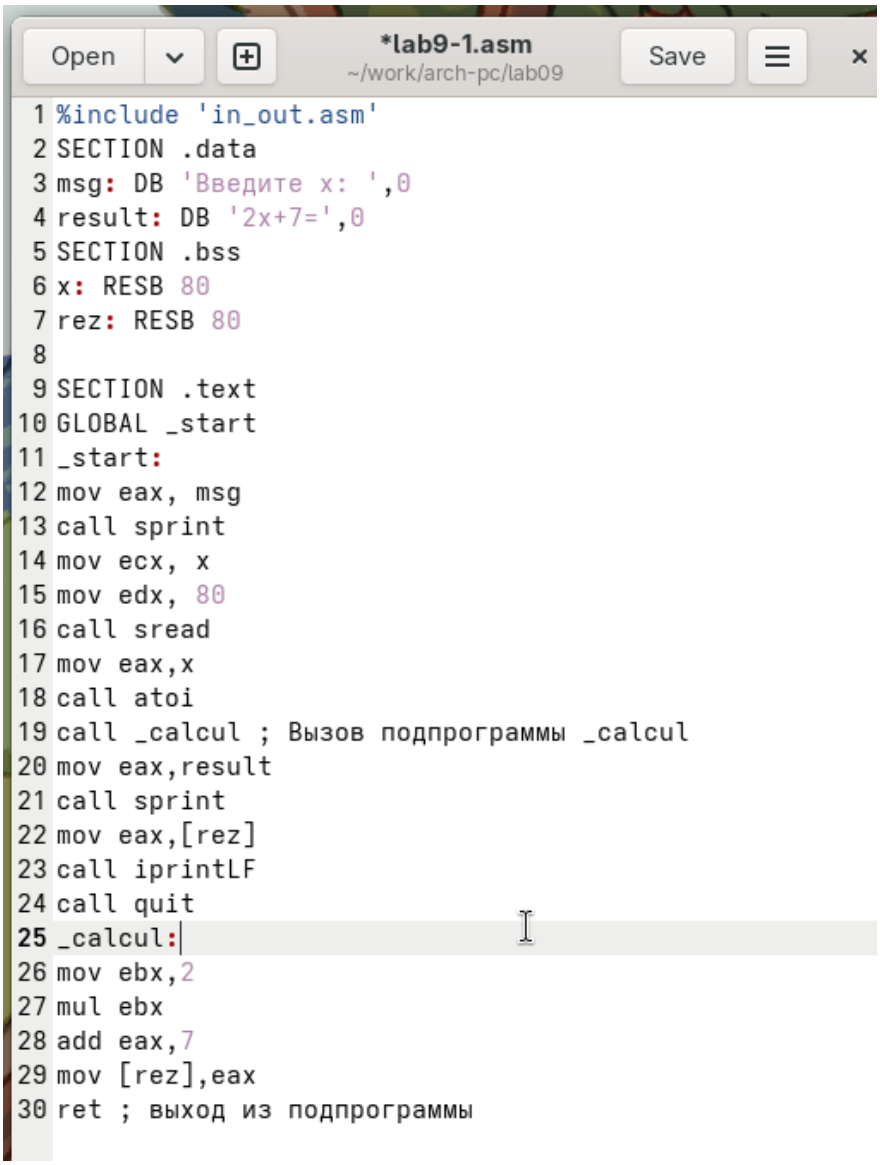

Рисунок 2.1: Текст программы lab9-1.asm

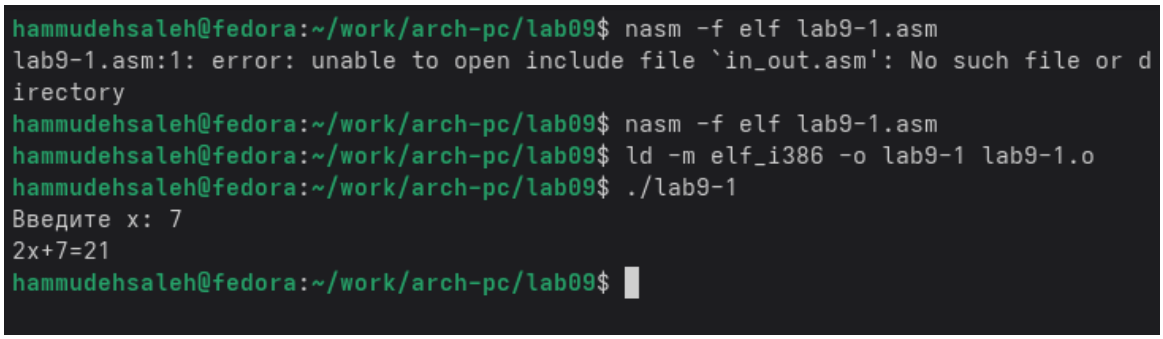

Рисунок 2.2: Запуск программы lab9-1.asm

Далее я модифицировал программу, добавив подпрограмму subcalcul внутрь подпрограммы calcul. Это позволило вычислять составное выражение $f(g(x))$, где $f(x) = 2x + 7$, а $g(x) = 3x - 1$. Значение x вводится с клавиатуры. (рис. 2.3) (рис. 2.4)

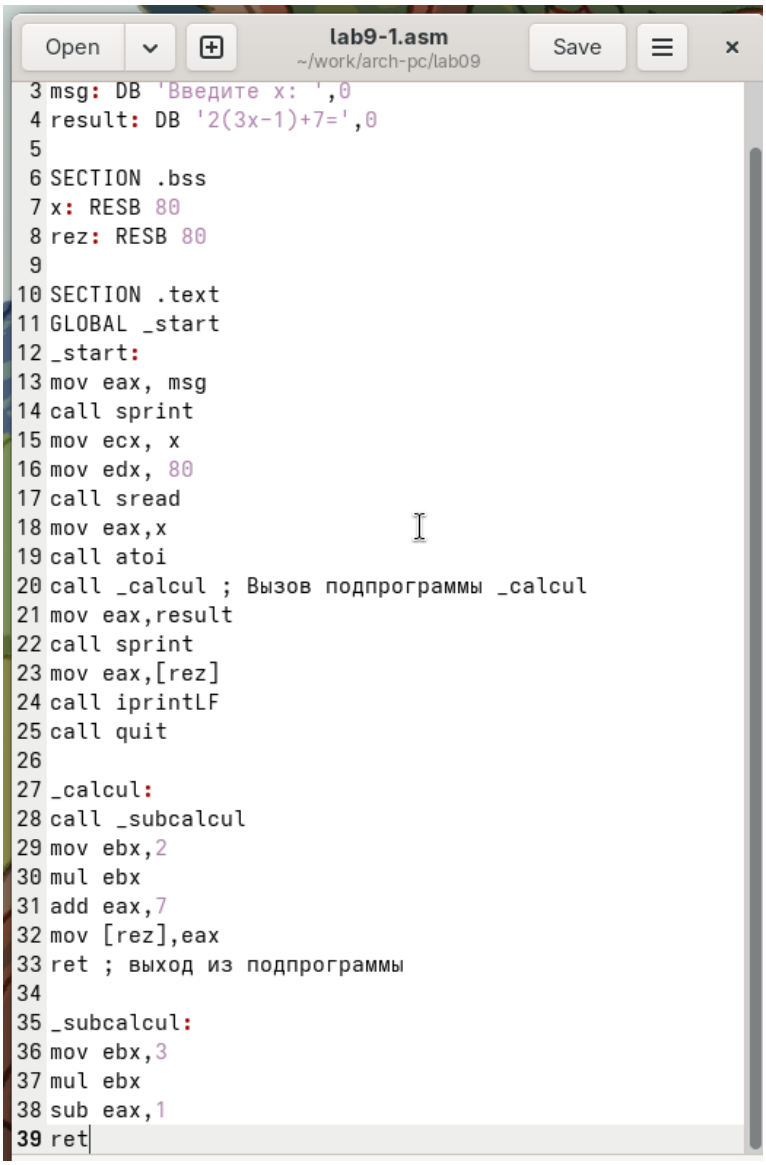

Рисунок 2.3: Модифицированная программа lab9-1.asm

```
hammudehsaleh@fedora:~/work/arch-pc/lab09$
hammudehsaleh@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
hammudehsaleh@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
hammudehsaleh@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
hammudehsaleh@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 7
2(3x-1)+7=47
hammudehsaleh@fedora:~/work/arch-pc/lab09$
```

Рисунок 2.4: 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. 2.5)

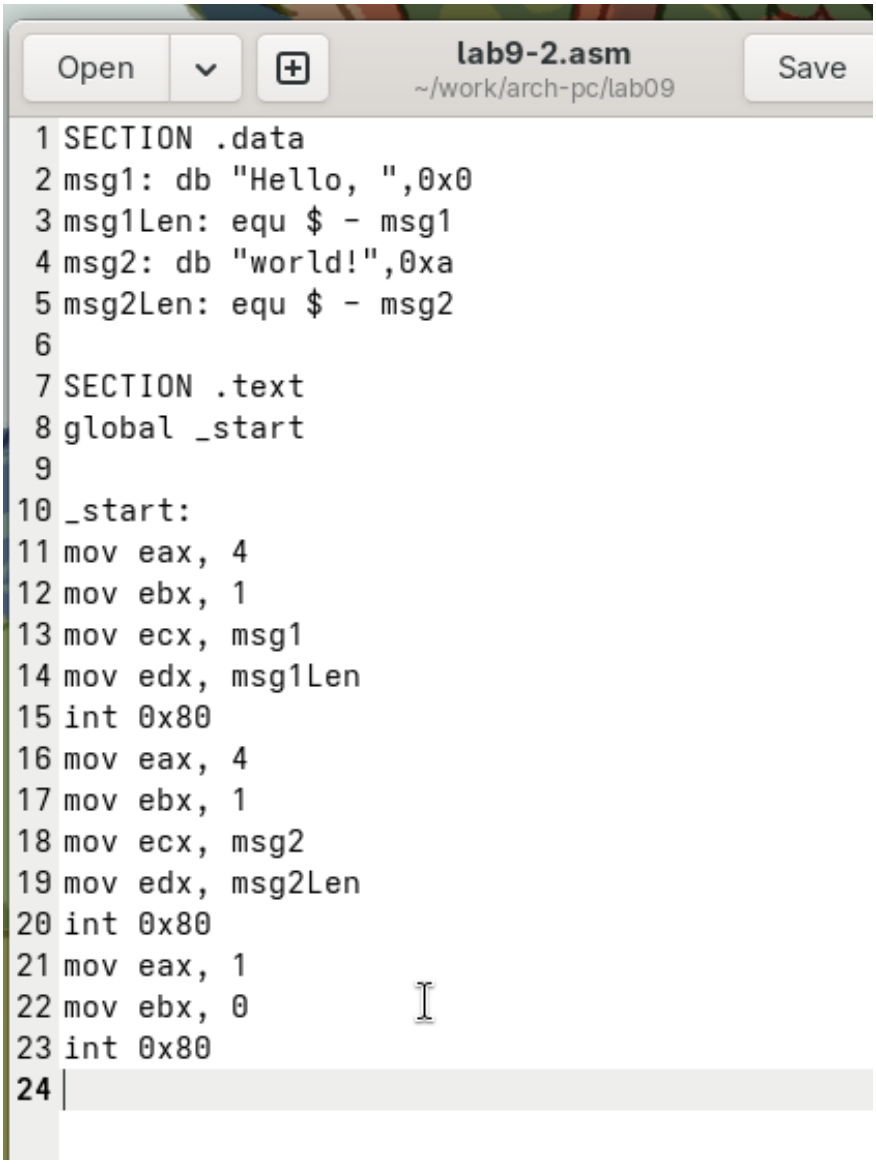

Рисунок 2.5: Код программы lab9-2.asm

После компиляции с ключом -g для добавления отладочной информации я загрузил исполняемый файл в GDB. Запустил программу с помощью команды run или r. (рис. 2.6)

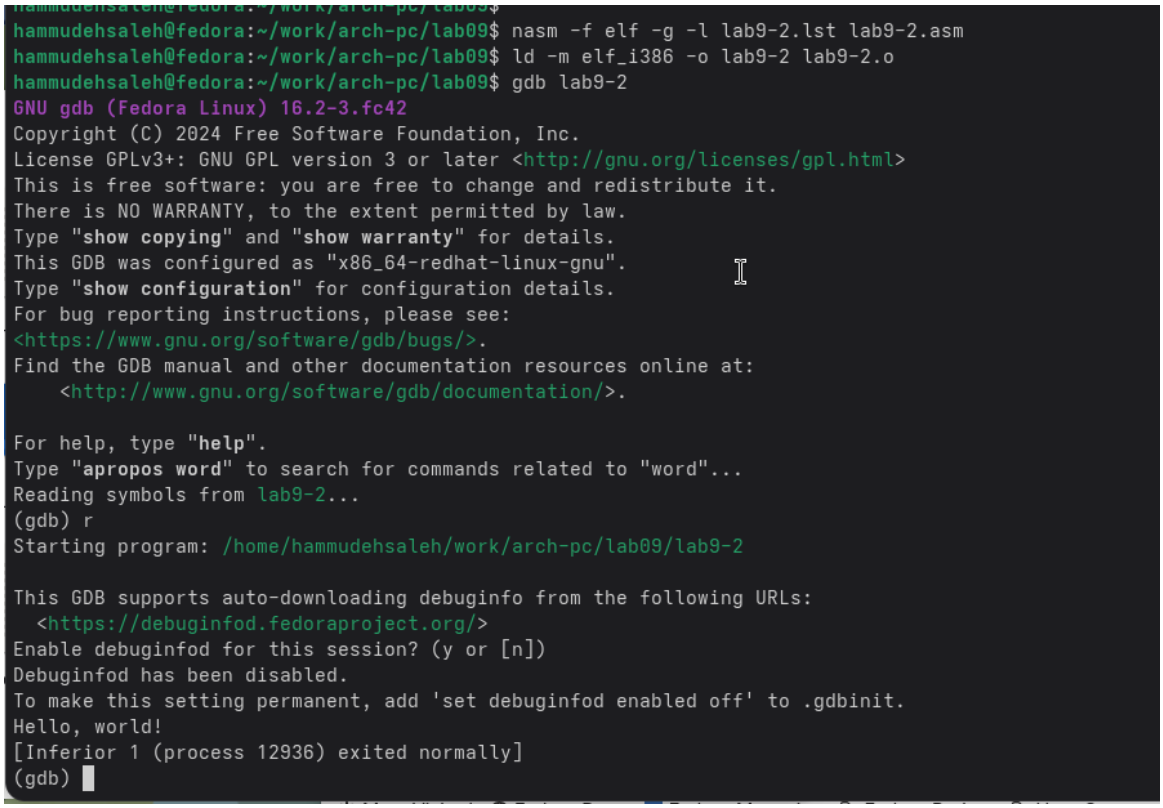

Рисунок 2.6: Запуск программы lab9-2.asm в GDB

Для анализа программы я установил точку остановки на метке _start и запустил выполнение. Затем изучил дизассемблированный код программы. (рис. 2.7) (рис. 2.8)

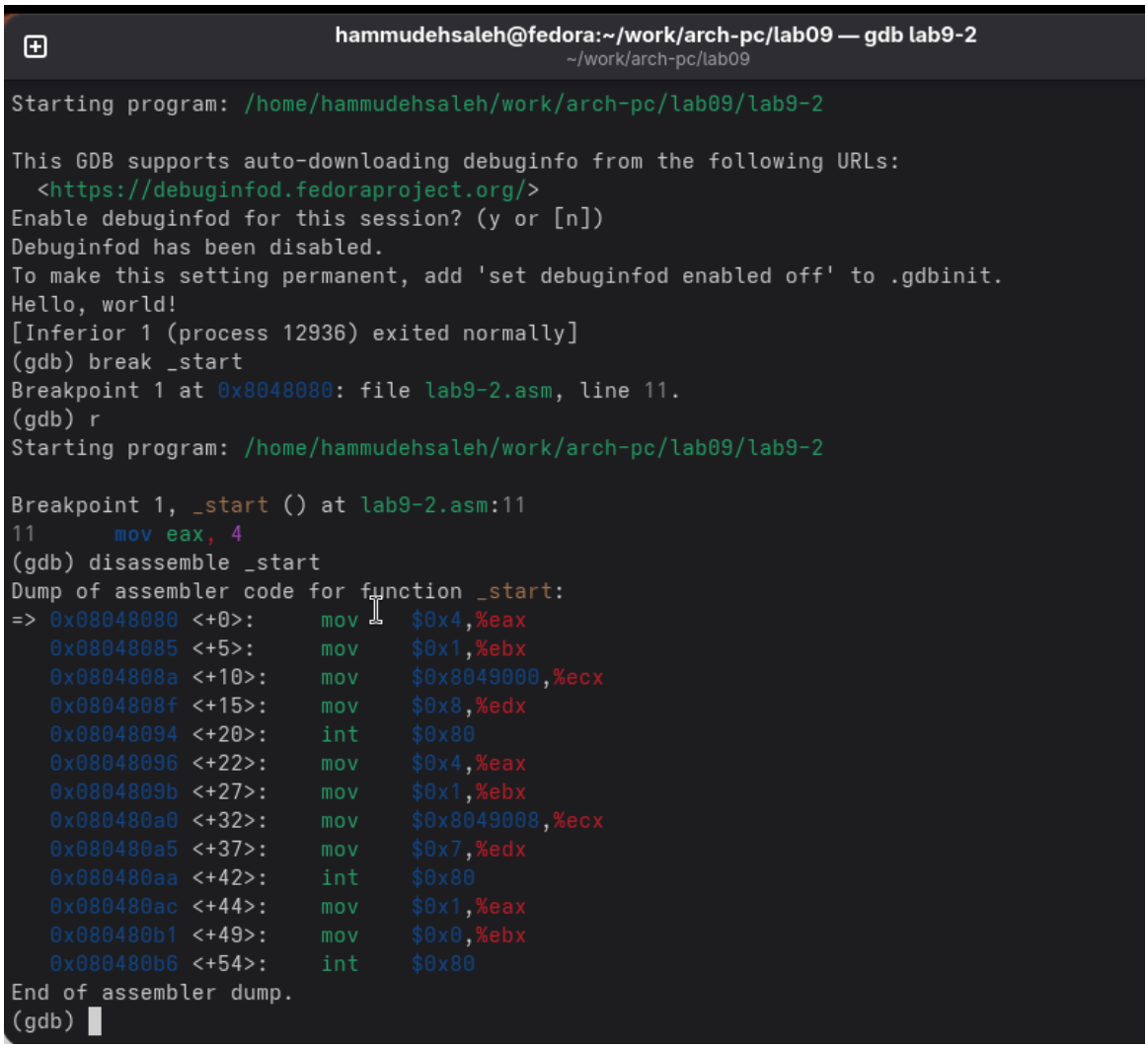

Рисунок 2.7: Дизассемблированный код программы

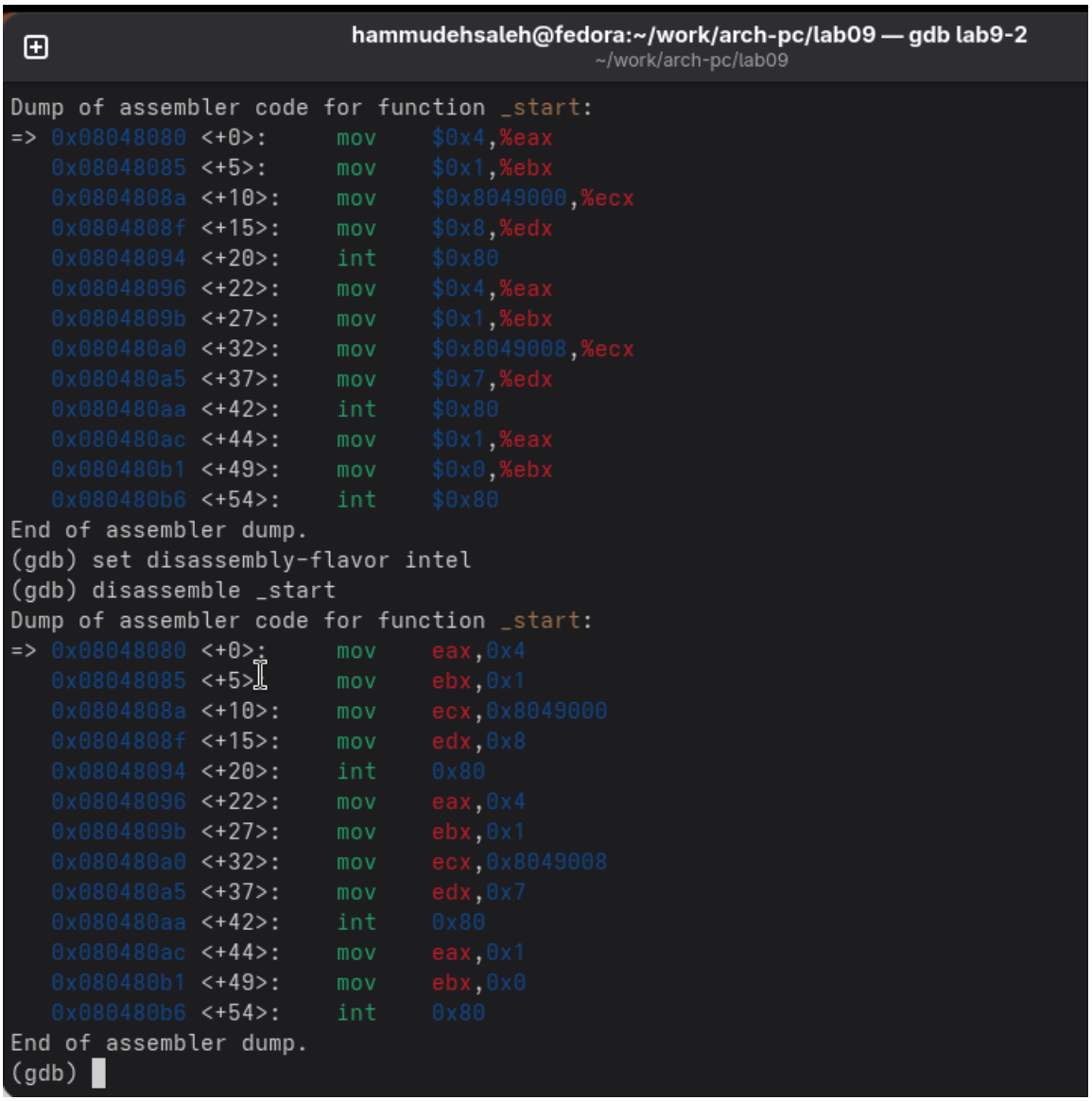

Рисунок 2.8: Дизассемблированный код в Intel-синтаксисе

Для проверки точки останова я использовал команду info breakpoints (i b). Установил дополнительную точку останова по адресу инструкции mov ebx, 0x0. (рис. 2.9)

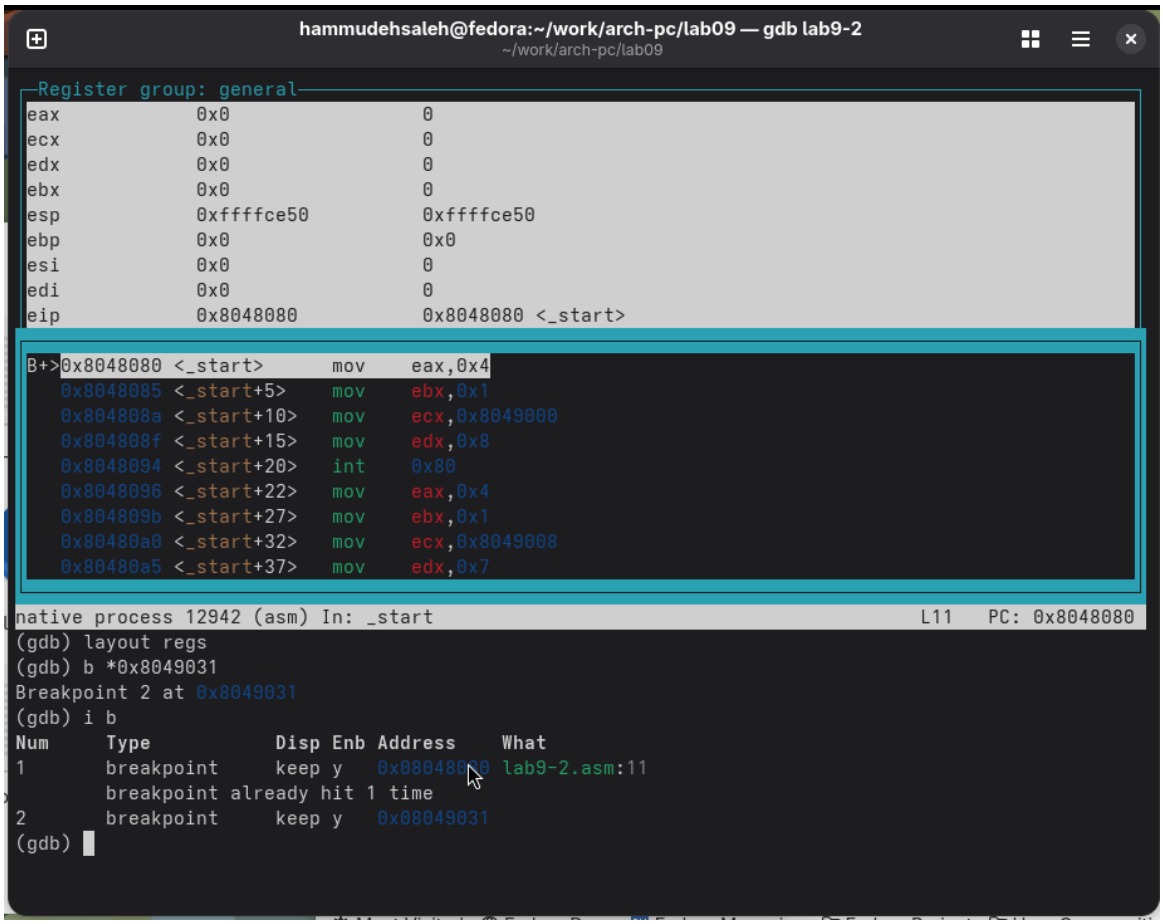

Рисунок 2.9: Настройка точки останова

С помощью команды stepi (si) выполнил пошаговую отладку, отслеживая изменения регистров. (рис. 2.10) (рис. 2.11)

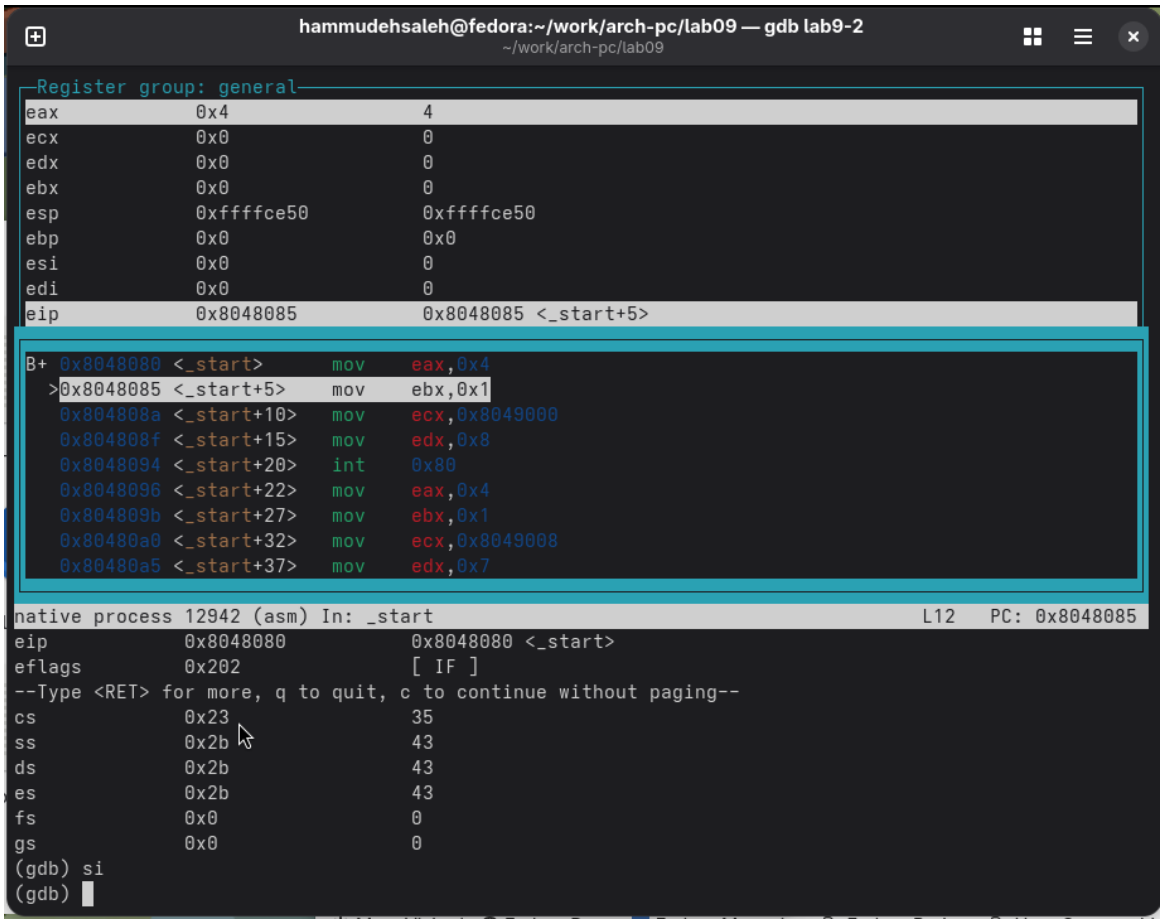

Рисунок 2.10: Отслеживание изменений регистров

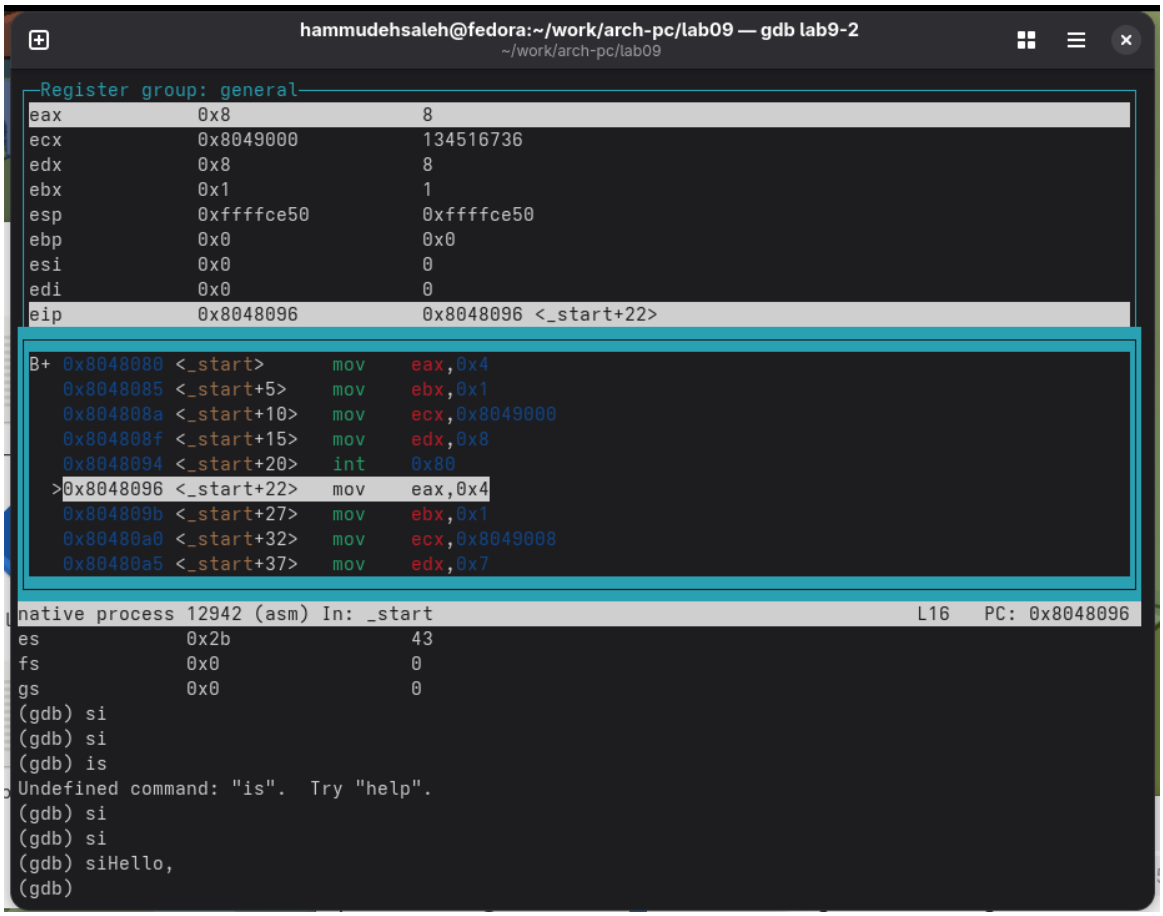

Рисунок 2.11: Детальный анализ регистров

Я также просмотрел значение переменной msg1 по имени и изменил первый символ переменной с помощью команды set. (рис. 2.12) (рис. 2.13)

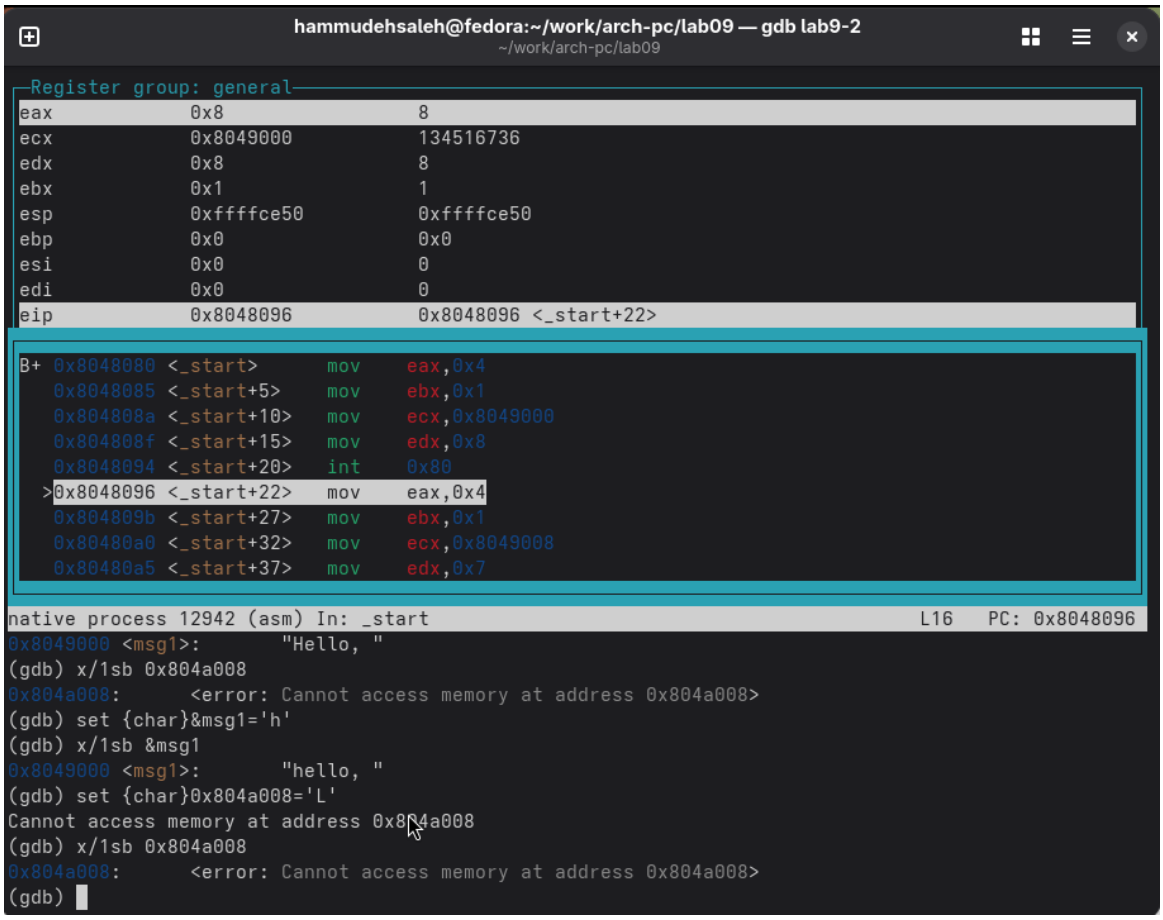

Рисунок 2.12: Изменение значения переменной msg1

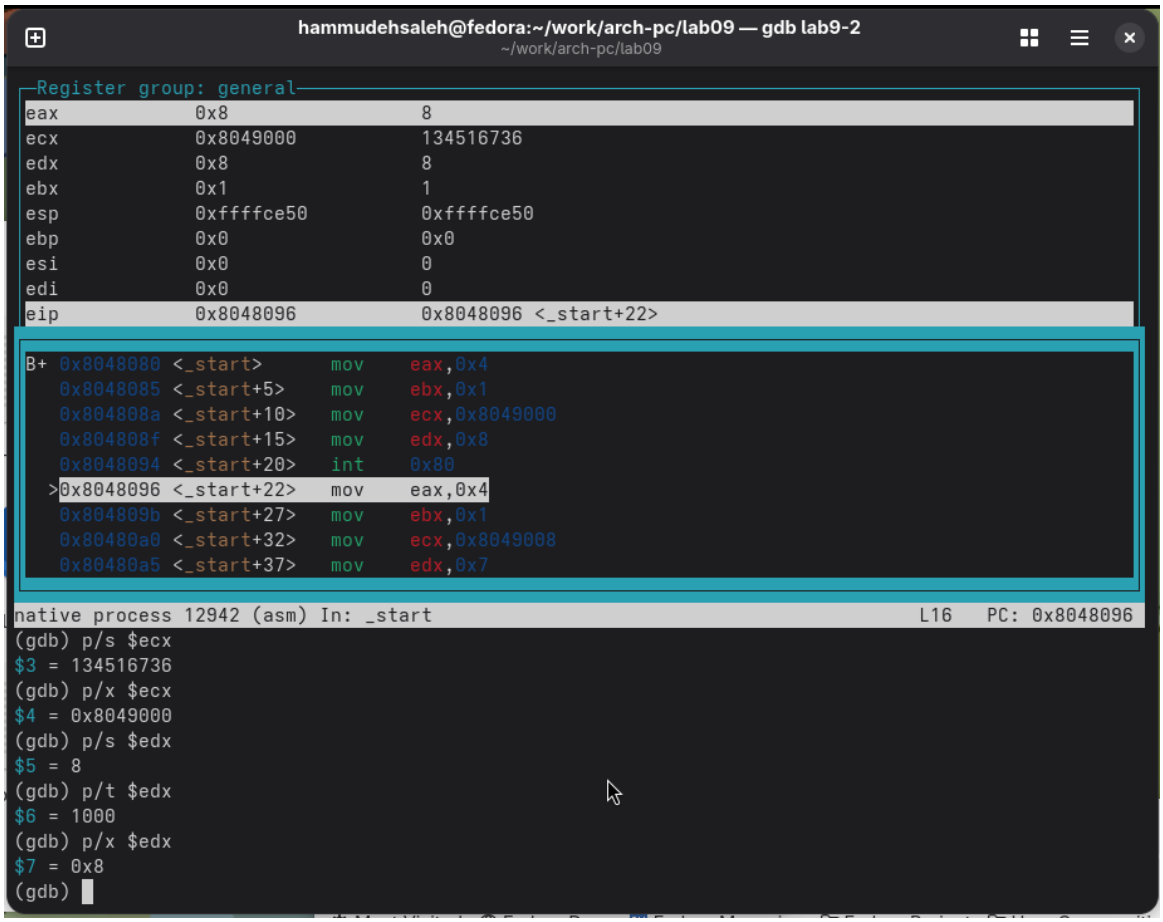

Рисунок 2.13: Просмотр регистра после изменений

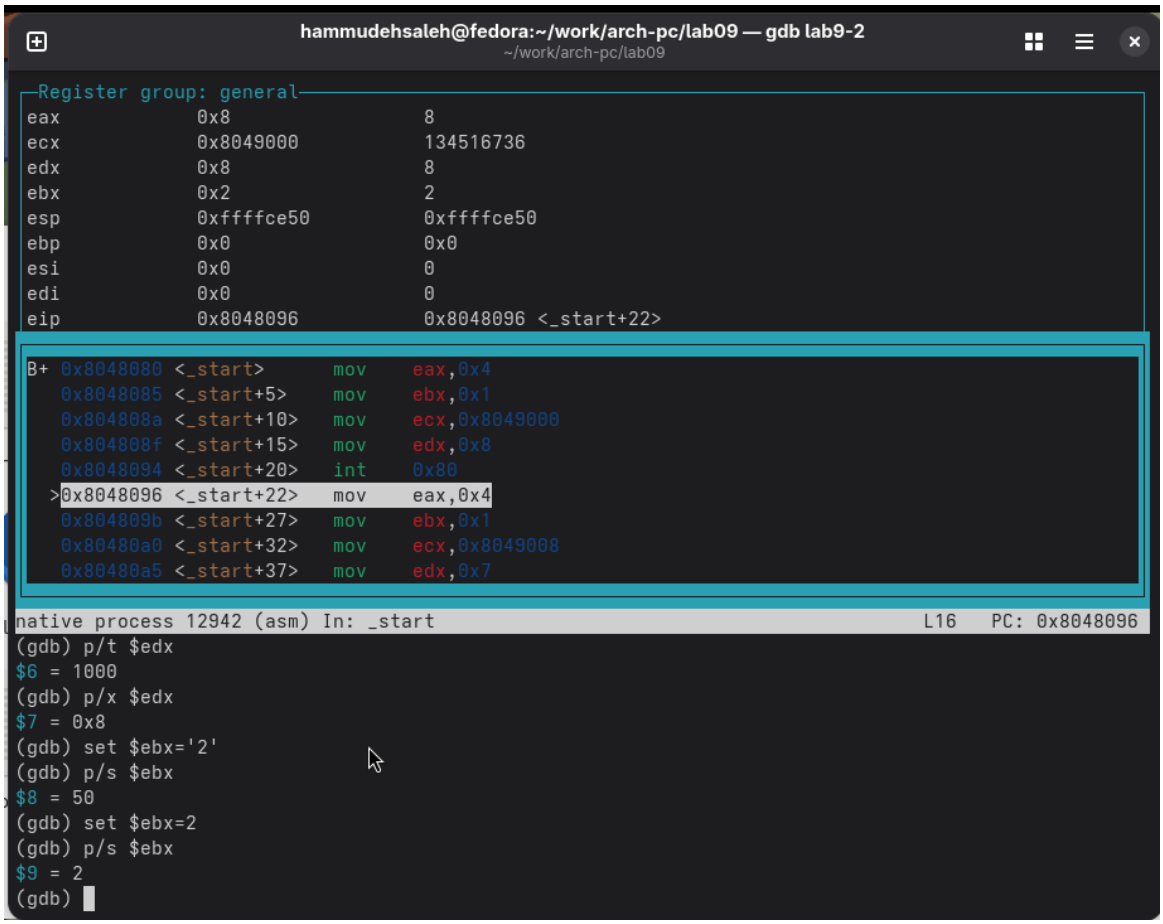

Рисунок 2.14: Изменение значения регистра

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. 2.15)

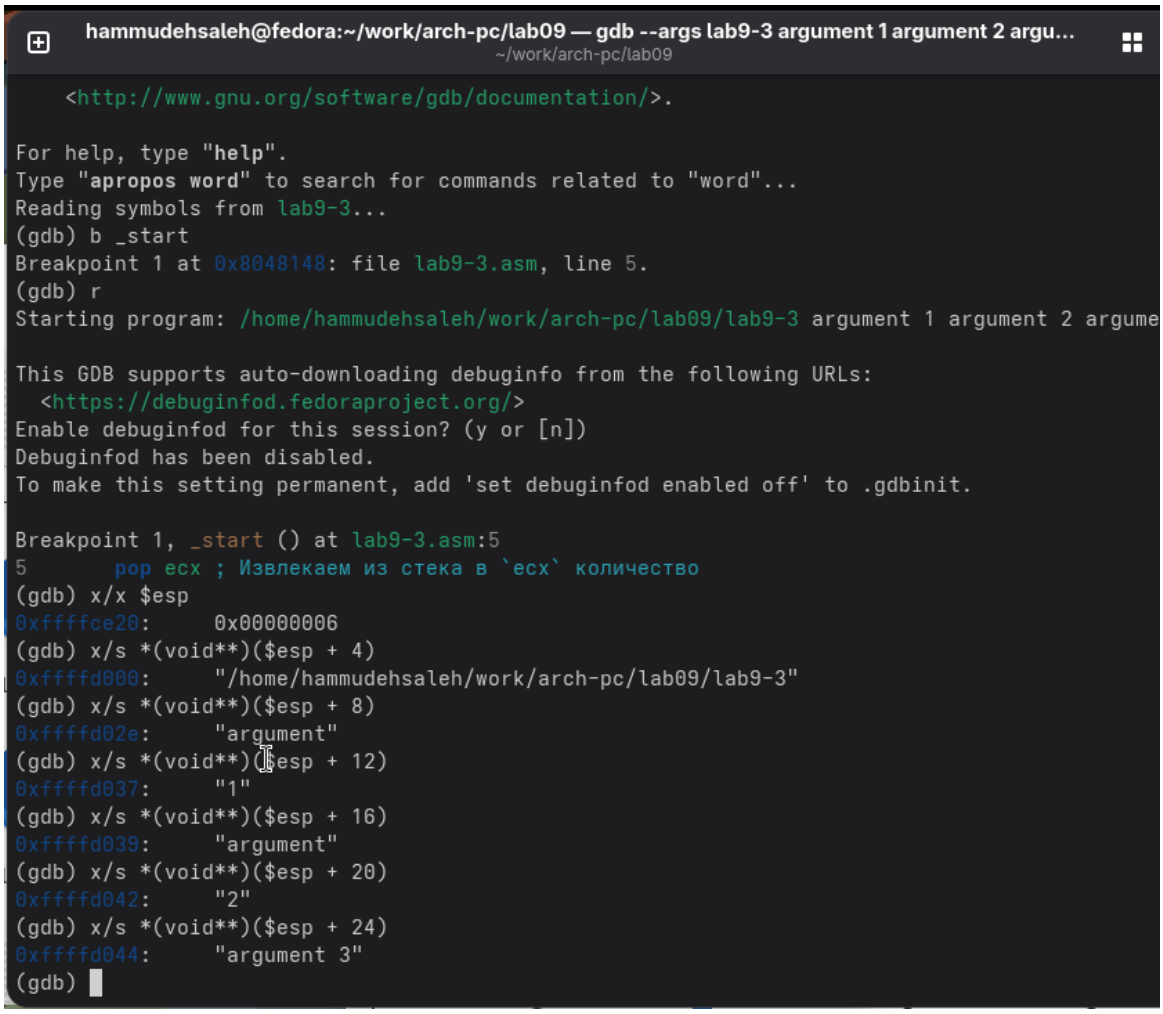

Рисунок 2.15: Анализ стека программы

2.3 Задание для самостоятельной работы

Я модифицировал программу из лабораторной работы №8, добавив вычисление функции $f(x)$ в виде подпрограммы. (рис. 2.16) (рис. 2.17)

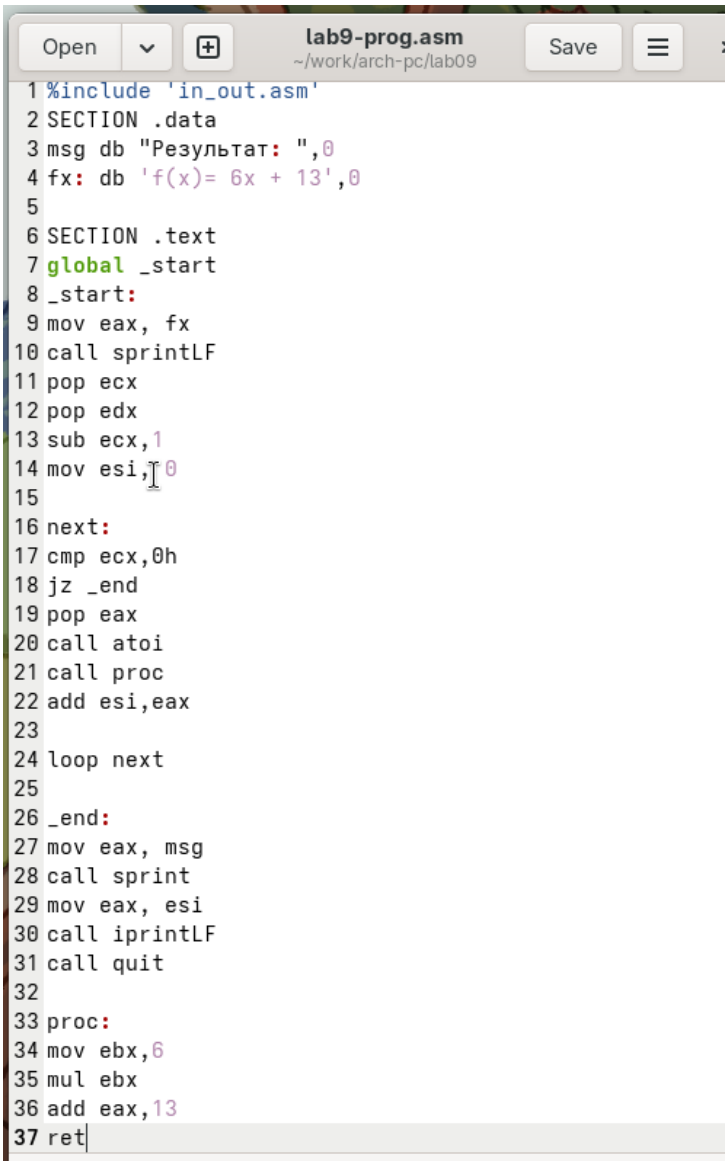

Рисунок 2.16: Код программы lab9-prog.asm

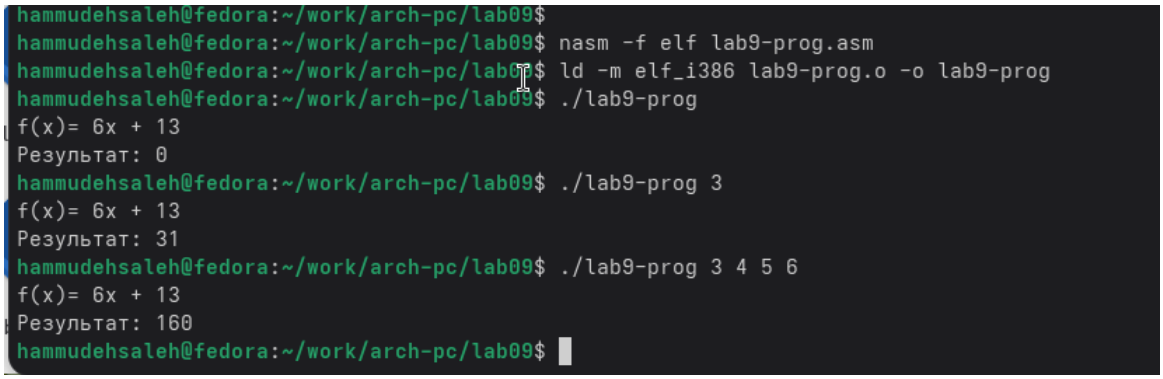

Рисунок 2.17: Запуск программы lab9-prog.asm

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. 2.18) (рис. 2.19)

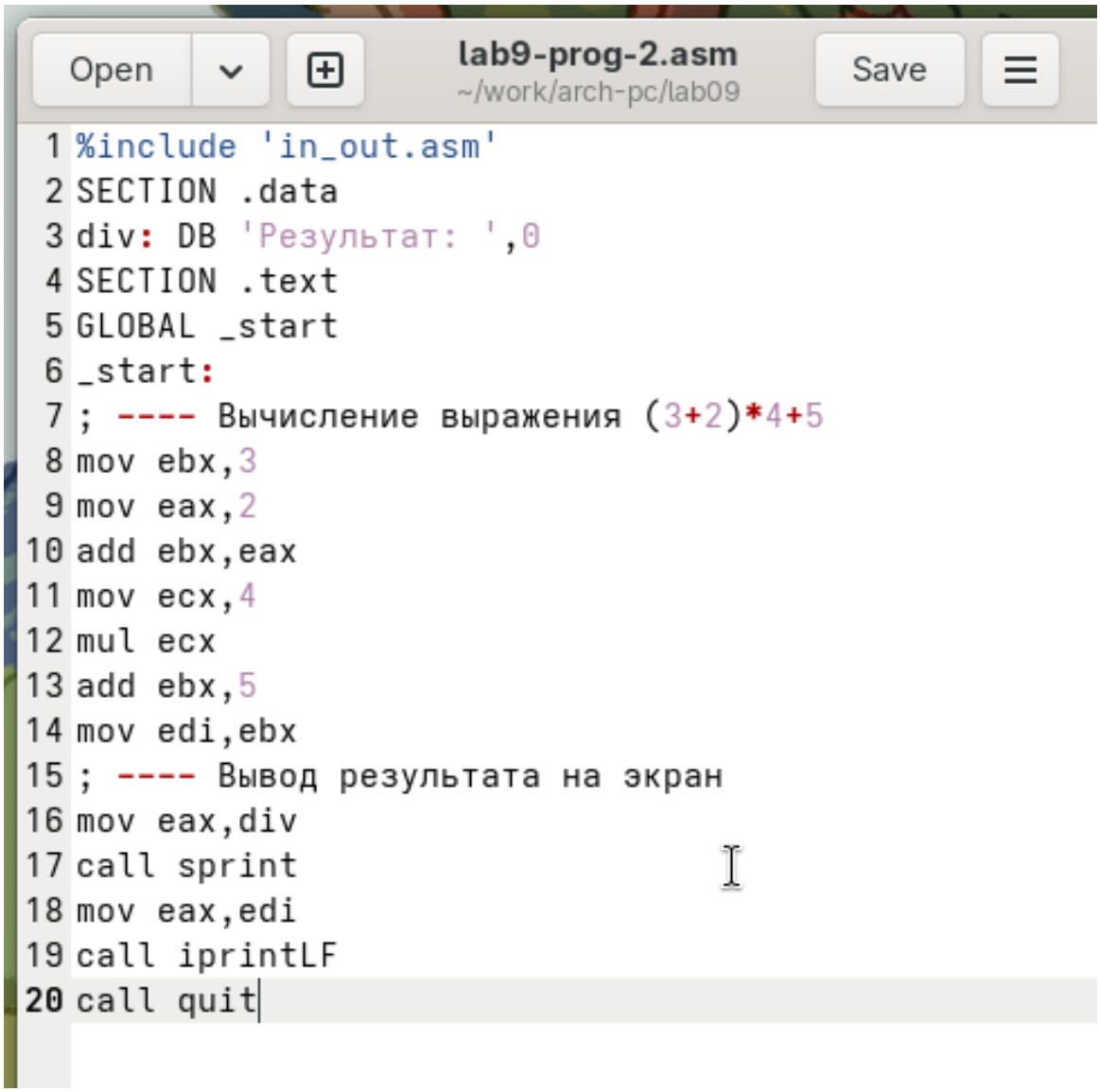

Рисунок 2.18: Код с ошибкой

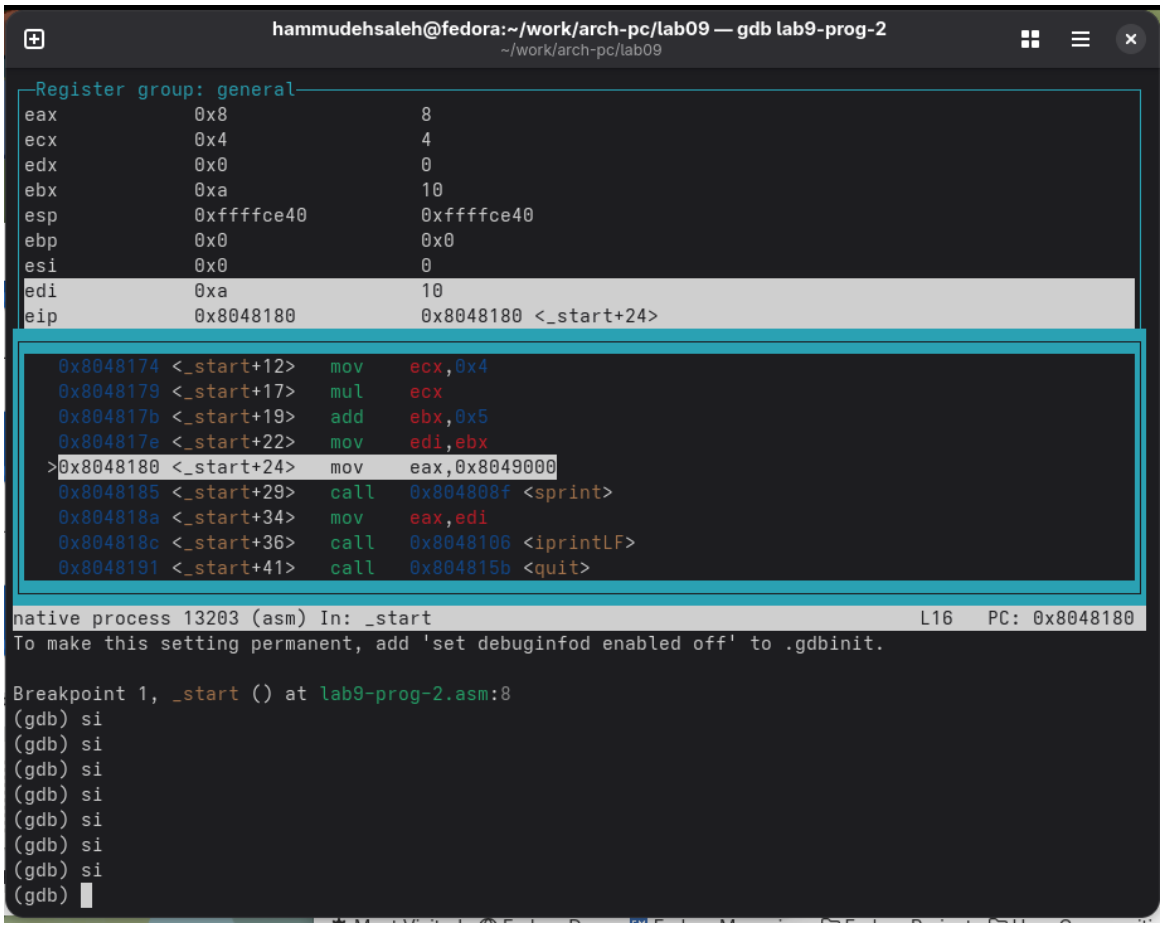

Рисунок 2.19: Процесс отладки программы

После исправления ошибок я проверил работу программы. (рис. 2.20) (рис. 2.21)

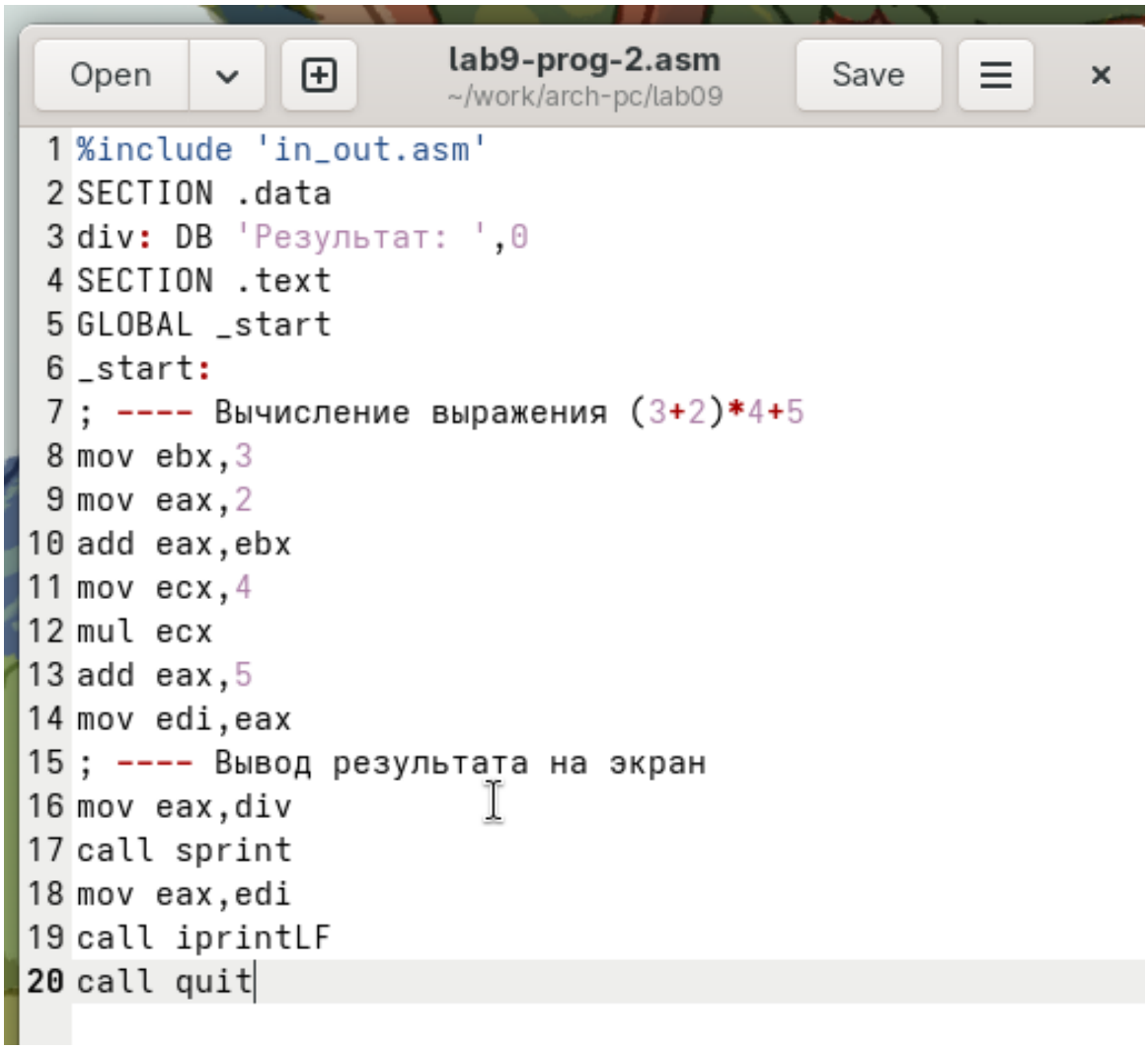

Рисунок 2.20: Исправленный код программы

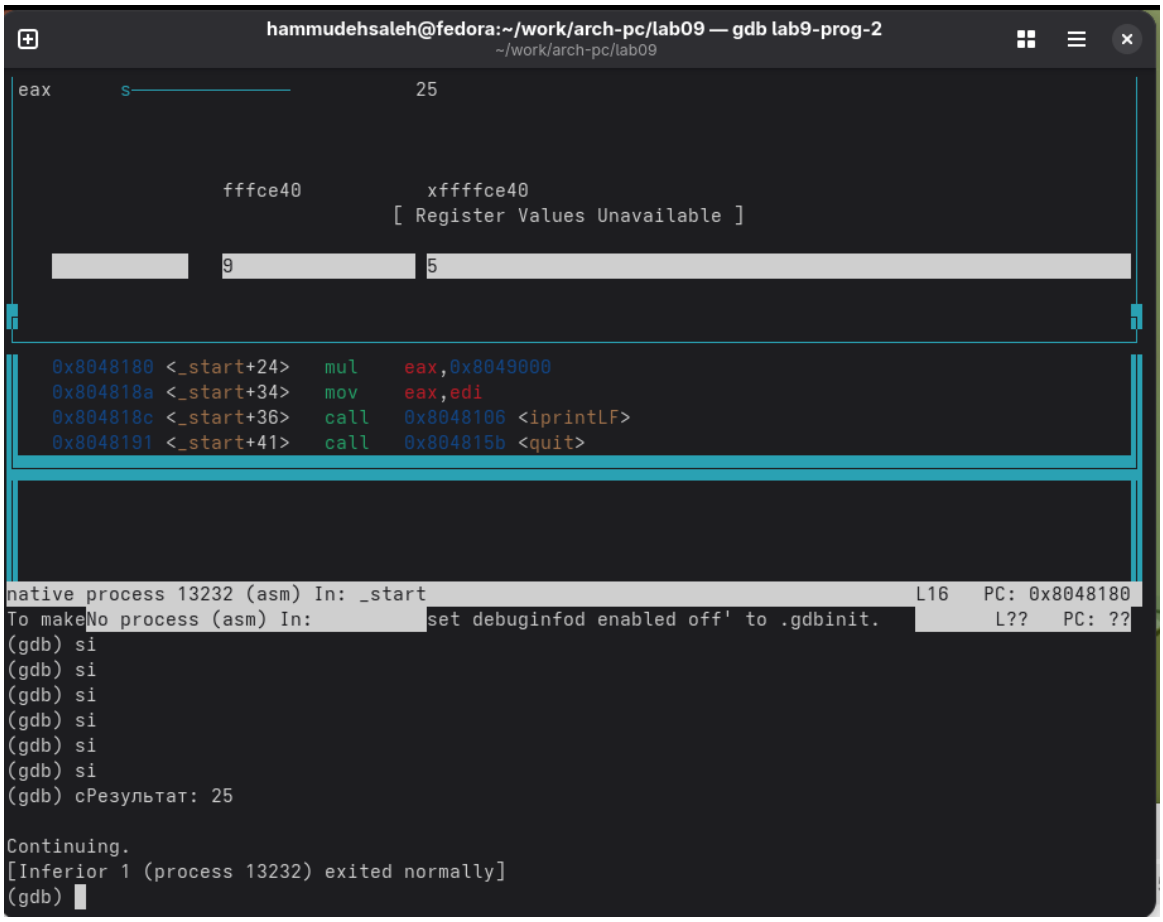

Рисунок 2.21: Проверка исправленного кода

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.