ОТЧЕТ по лабораторной работе №9

Полякова Юлия Александровна

Содержание

# 1 Цель работы

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

# 2 Результаты выполнения лабораторной работы

1. Создаем каталог для выполнения лабораторной работы № 9, переходим в него и создаем файл lab09-1.asm. Вводим в файл lab09-1.asm текст программы из листинга 9.1. (Рис. 1).

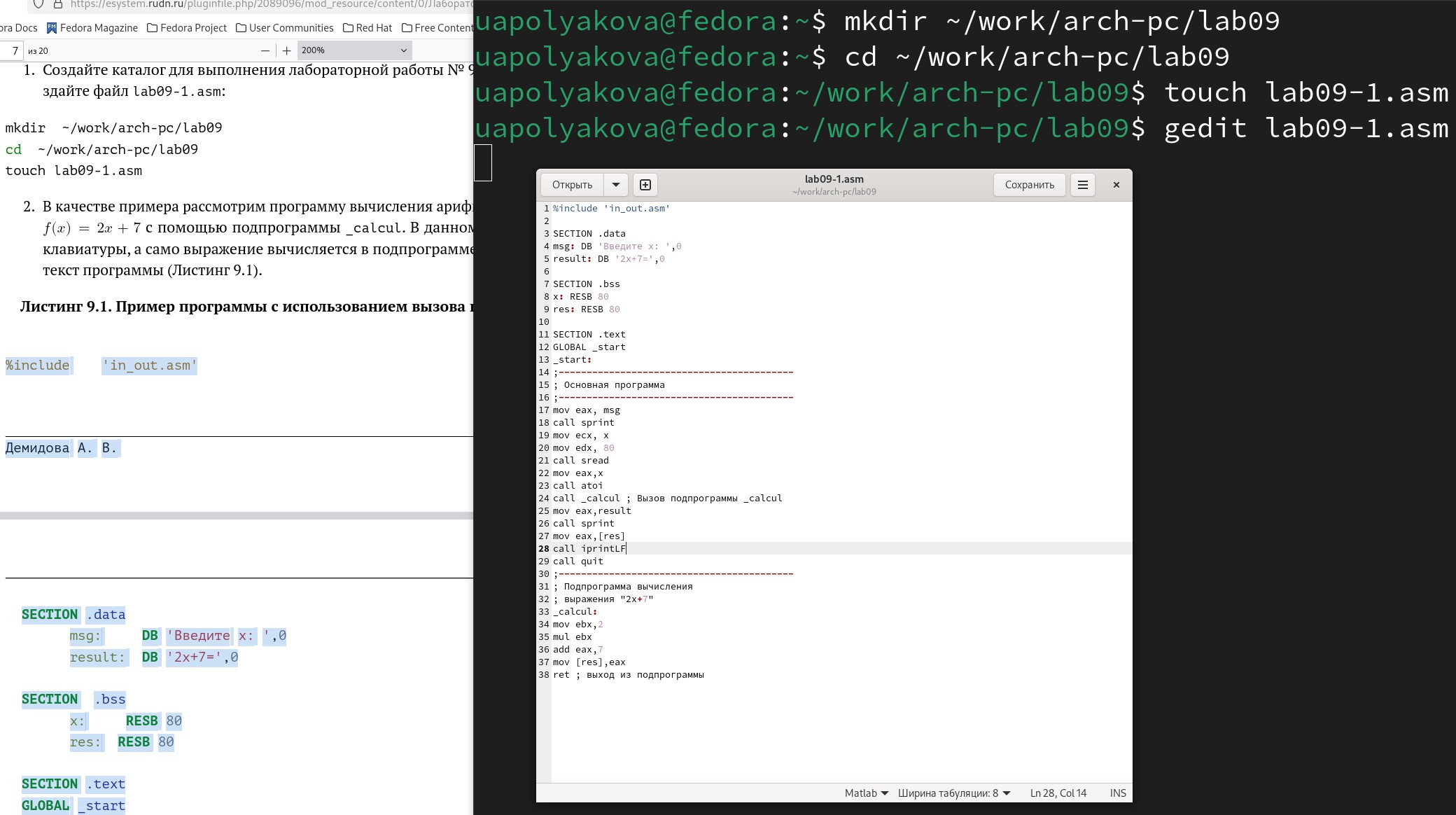


Рис. 1: Создание каталога и lab09-1.asm

1. Создаем исполняемый файл и запускаем его (Рис. 2).

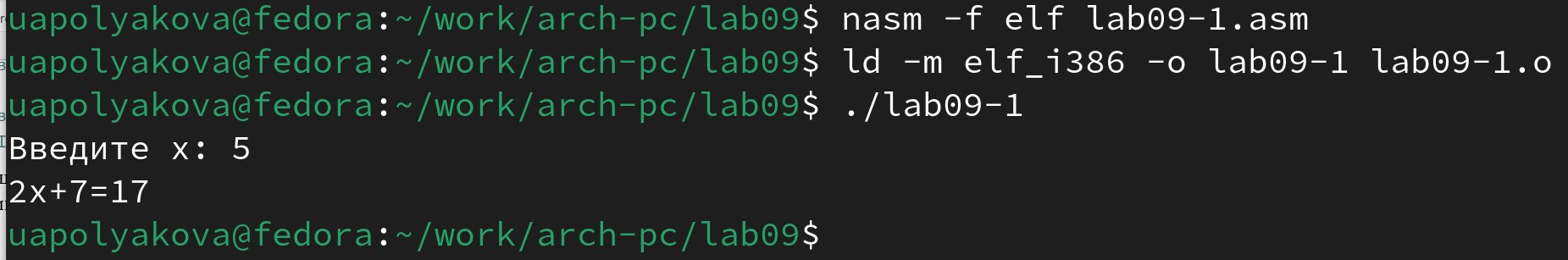


Рис. 2: Запуск lab09-1.asm из листинга 9.1.

1. Изменяем текст программы, чтобы вычисление g(x) было в \_calcul (Рис. 3).

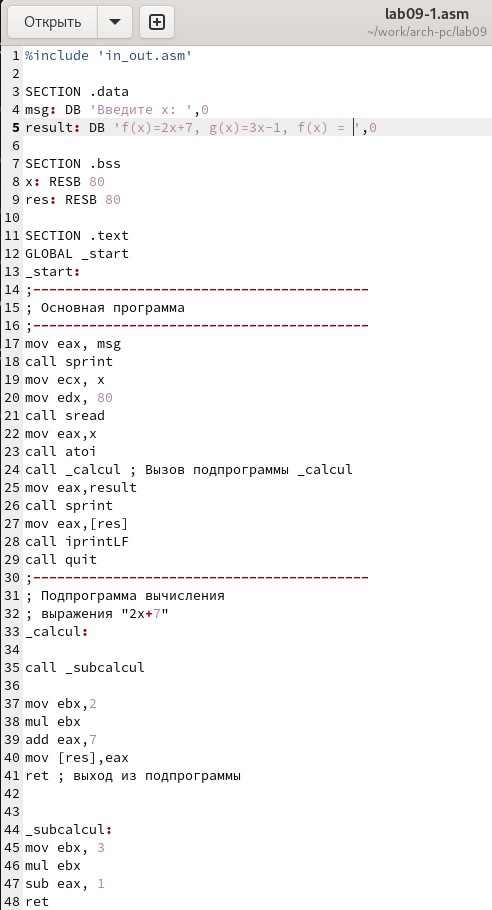


Рис. 3: Измененный lab09-1.asm

1. Создаем исполняемый файл и запускаем его (Рис. 4)

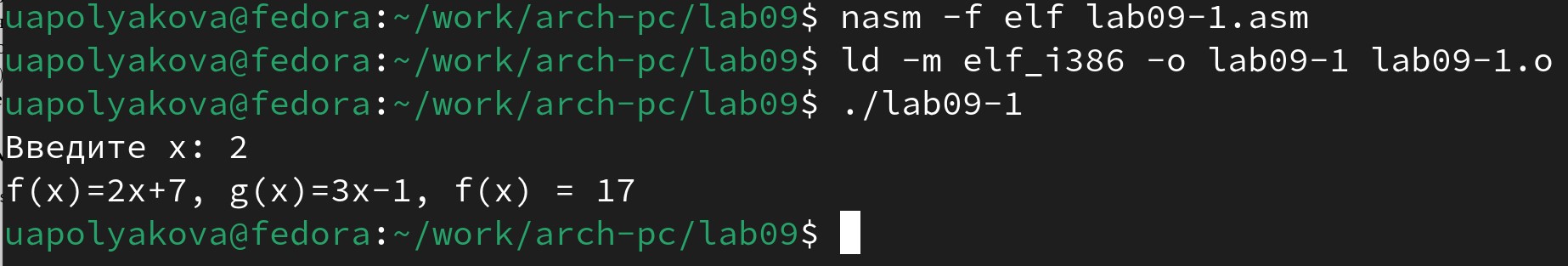


Рис. 4: Запукс измененного lab09-1.asm

1. Создаем файл lab09-2.asm с текстом программы из Листинга 9.2. (Рис. 5).

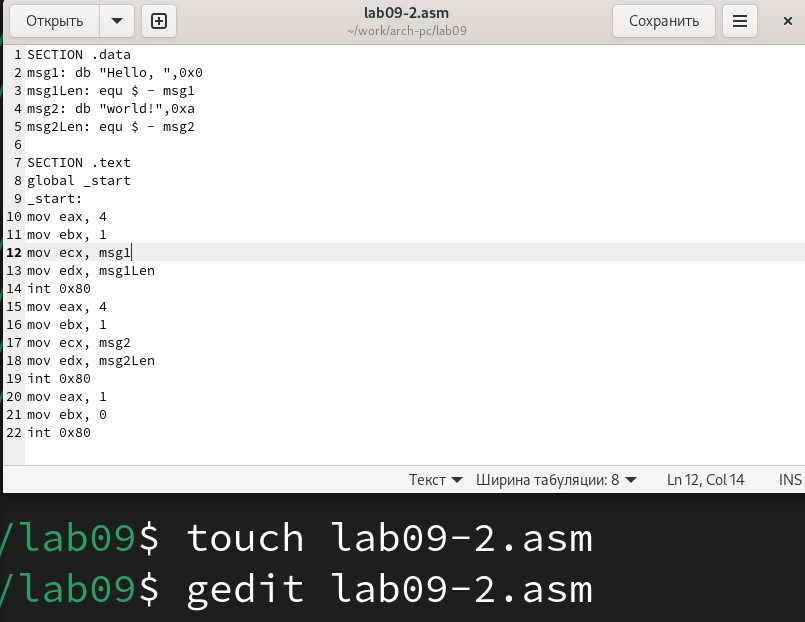


Рис. 5: lab09-2.asm

1. Транслируем с аргументом отладки, загружаем фал в отладчик (Рис. 6).

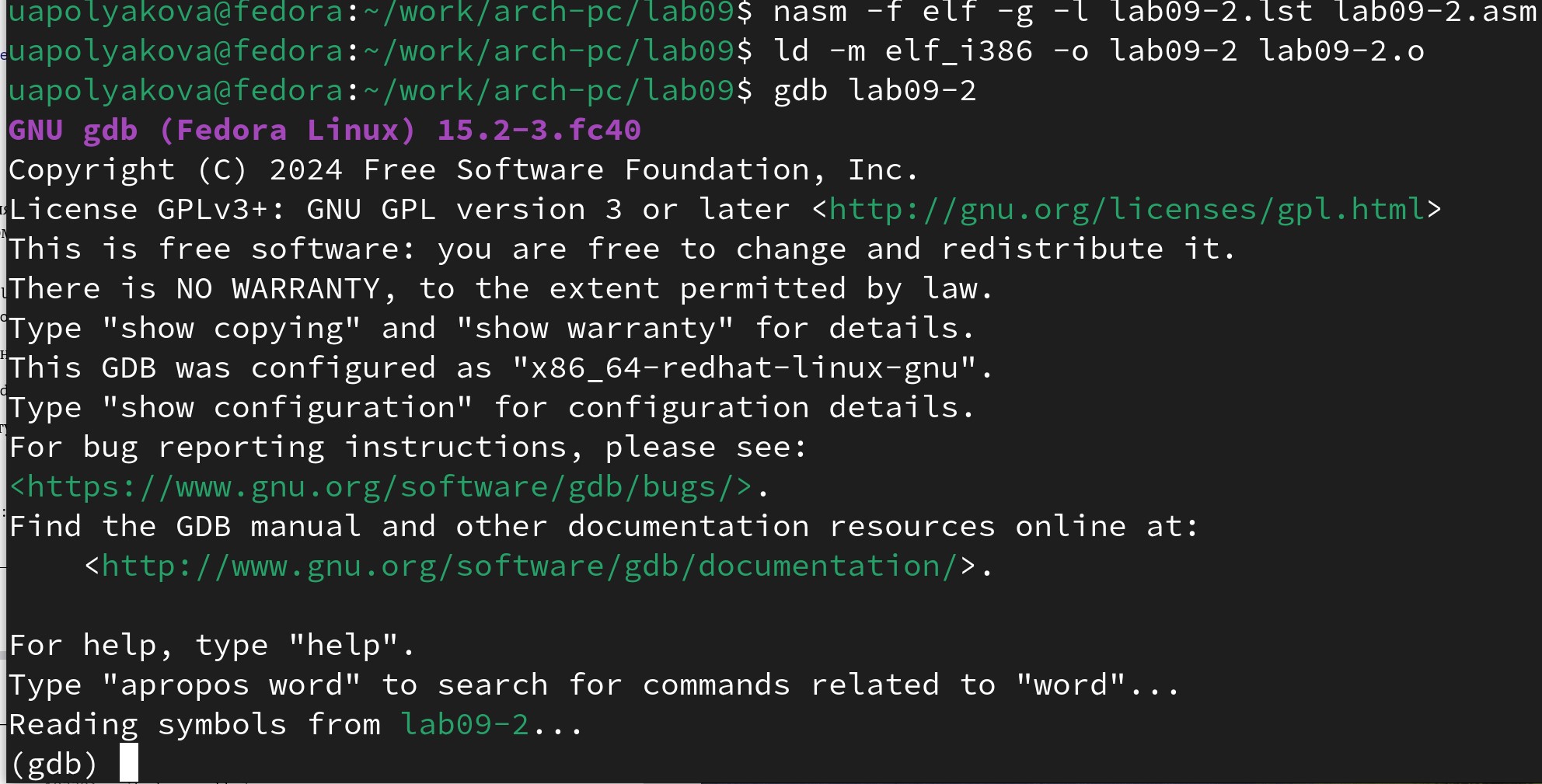


Рис. 6: Загрузка файла в отладчик

1. Запускаем программу командой run (Рис. 7).

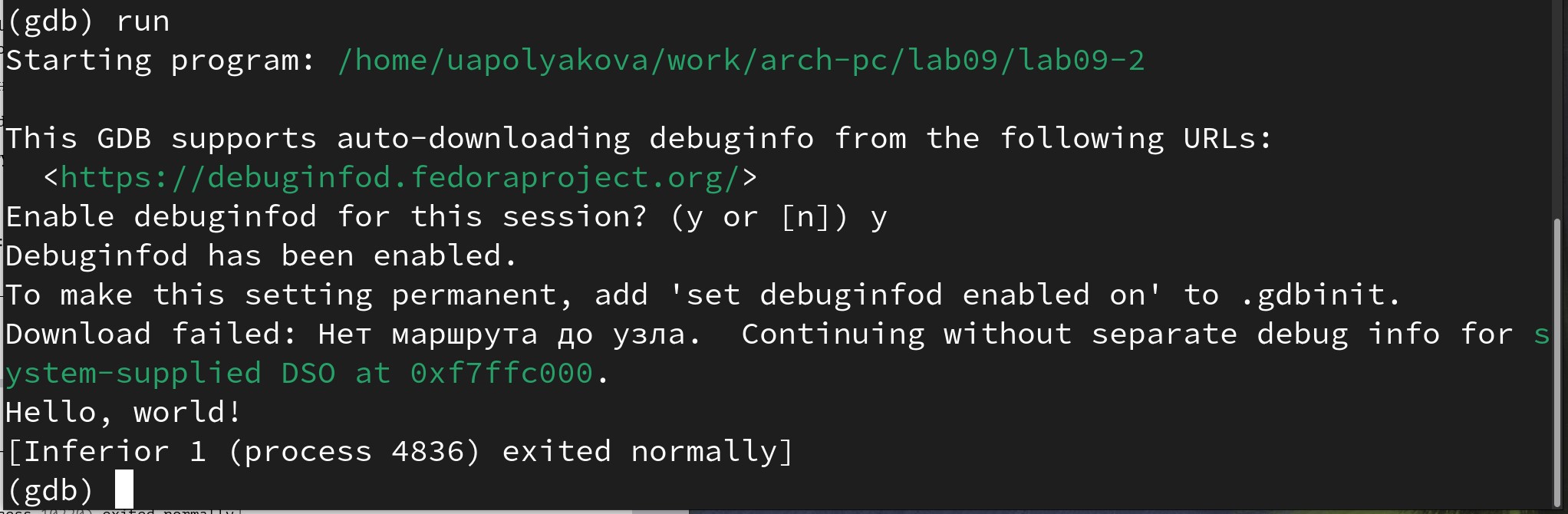


Рис. 7: Запуск командой run

1. Ставим брейкпоинт на \_start (Рис. 8).

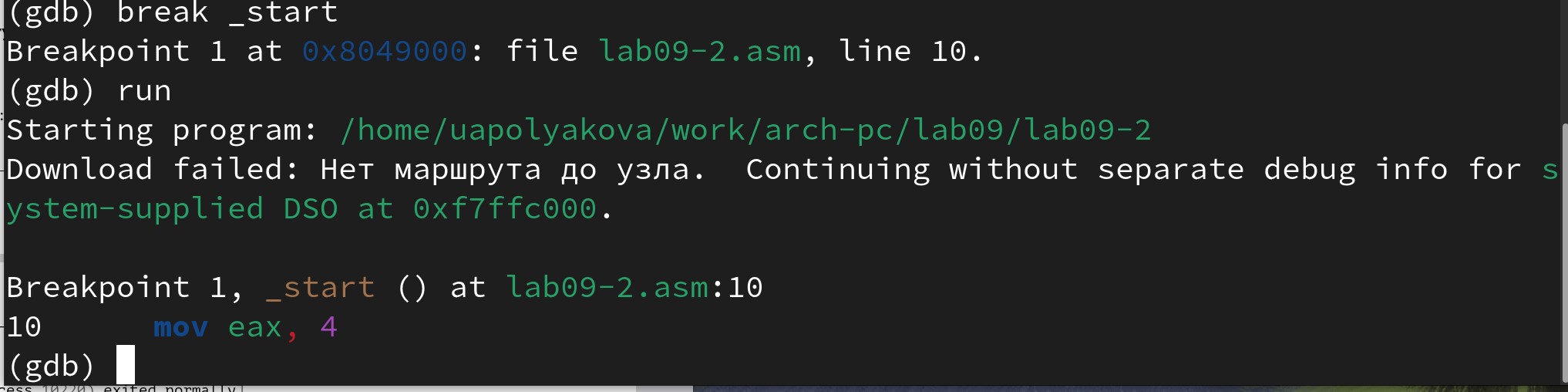


Рис. 8: Ставим брейкпоинт

1. Смотрим дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (Рис. 9)

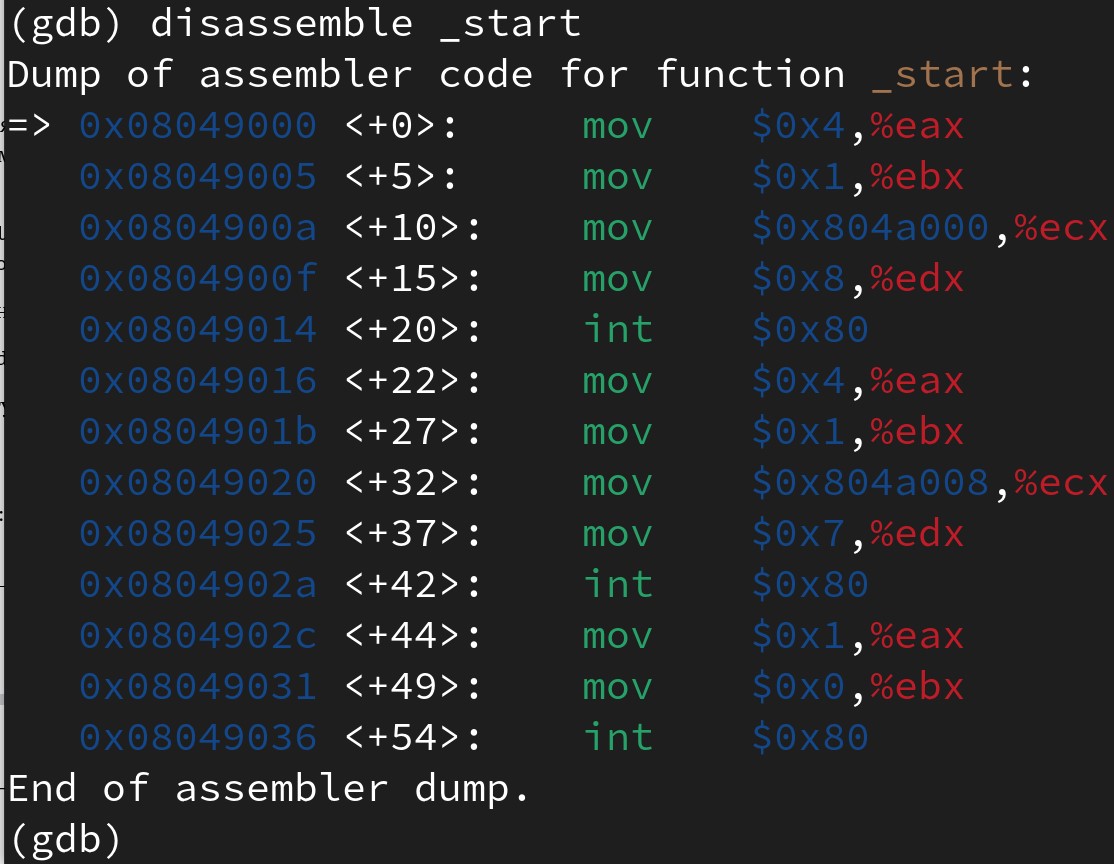


Рис. 9: Дисассимилированный код

1. Переключаемся на отображение команд с Intel’овским синтаксисом (Рис. 10).



Рис. 10: Переключаемся на Intel’овский синтаксис

В Intel’овском синтаксисе регистр идет первым аргументом и опускаются знаки ($, %)

1. Включаем режим псевдографики (Рис. 11).

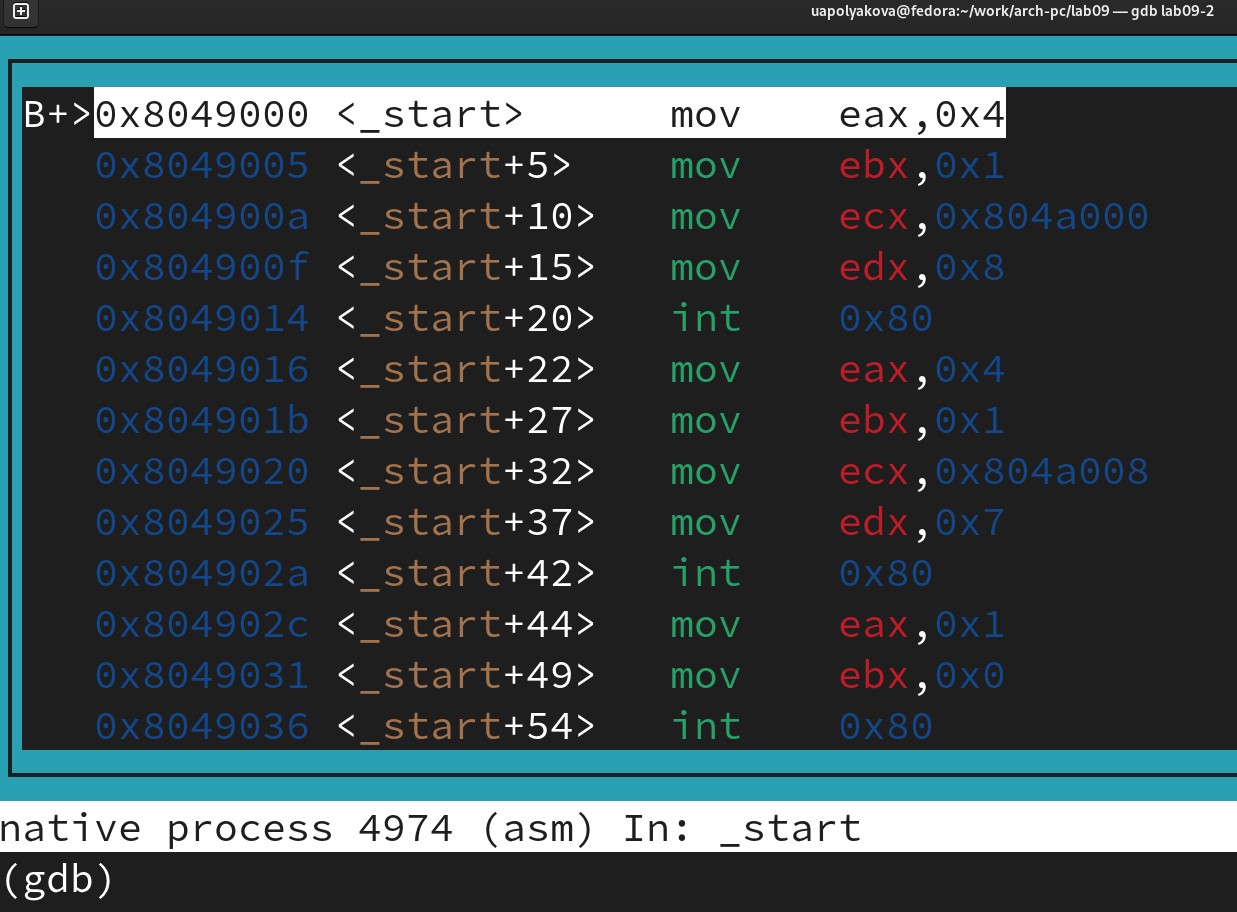


Рис. 11: Режим псевдографики

1. Включаем окно с регистрами (Рис. 12).

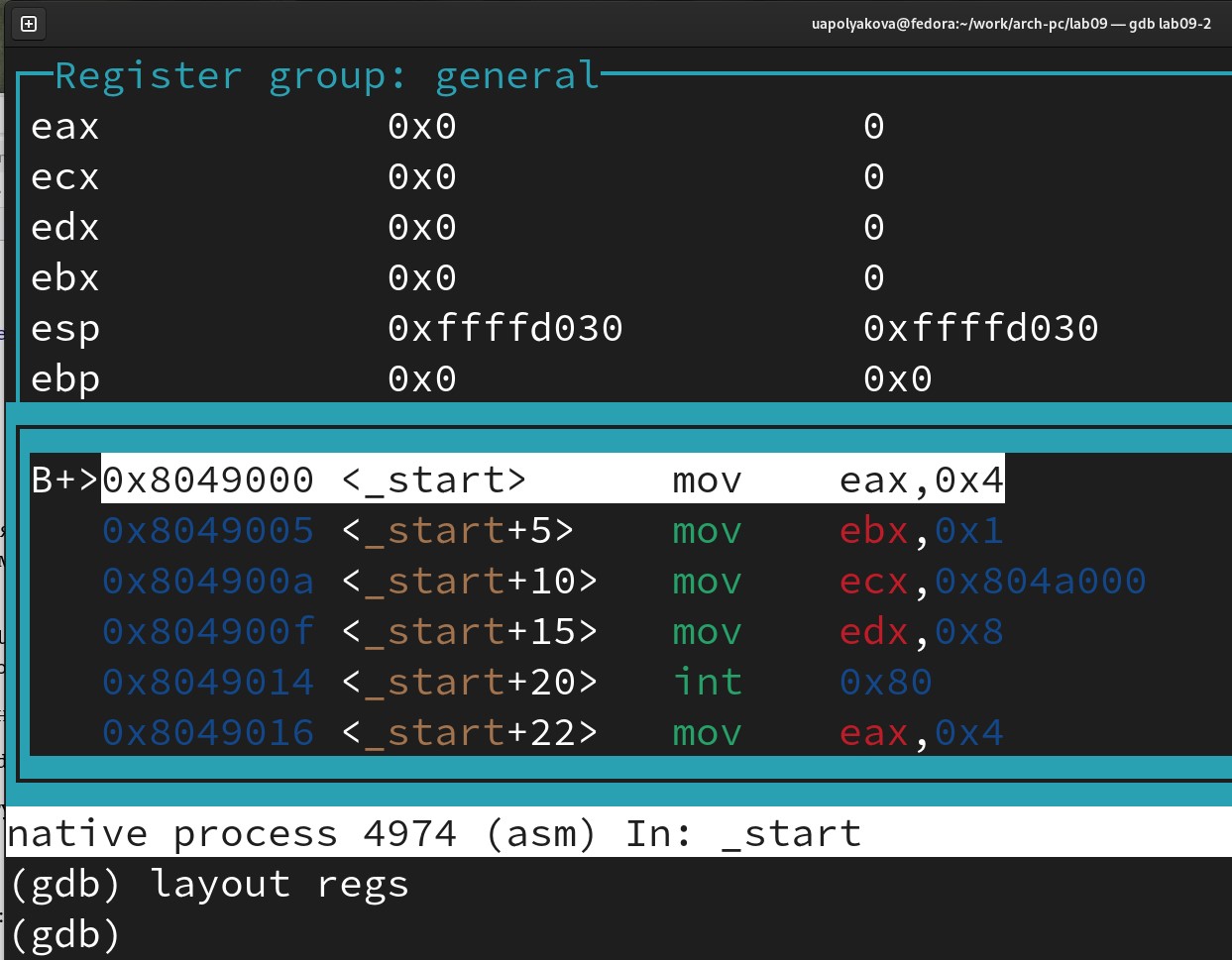


Рис. 12: Окно с регистрами

1. Проверяем информацию о установленном ранее брейкпоинте (Рис. 13).

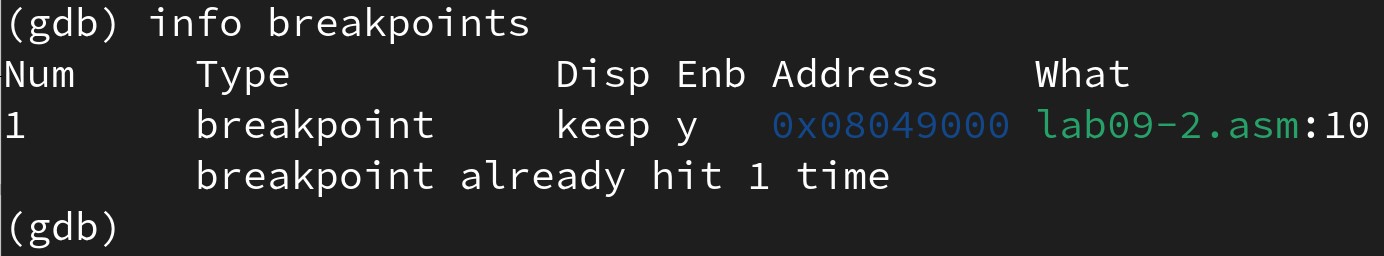


Рис. 13: Проверяем информацию о брейкпоинте

1. Устанавливаем брейкпоинт по адресу и проверяем его (Рис. 14).

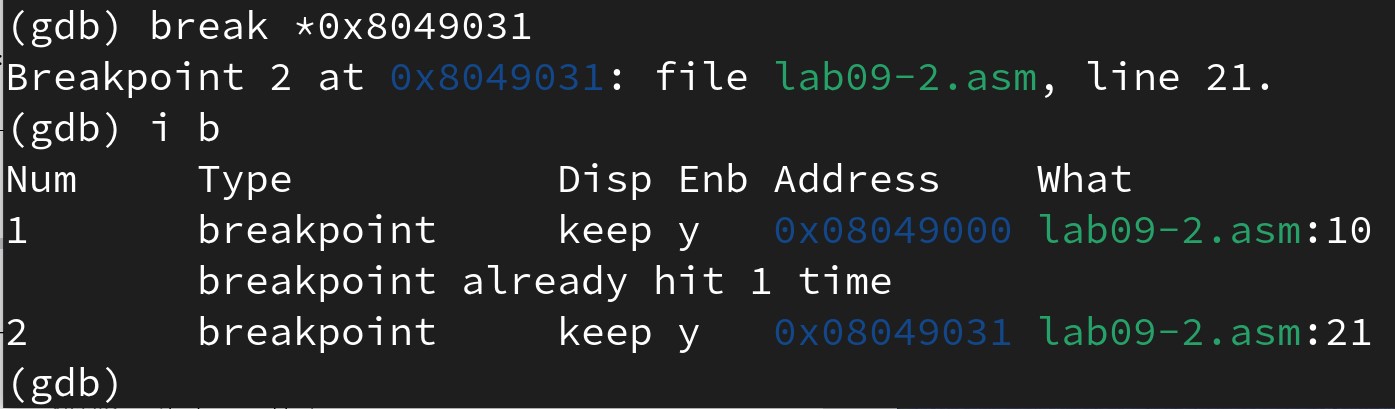


Рис. 14: Брейкпоинт по адресу

1. Первое si (Рис. 15).

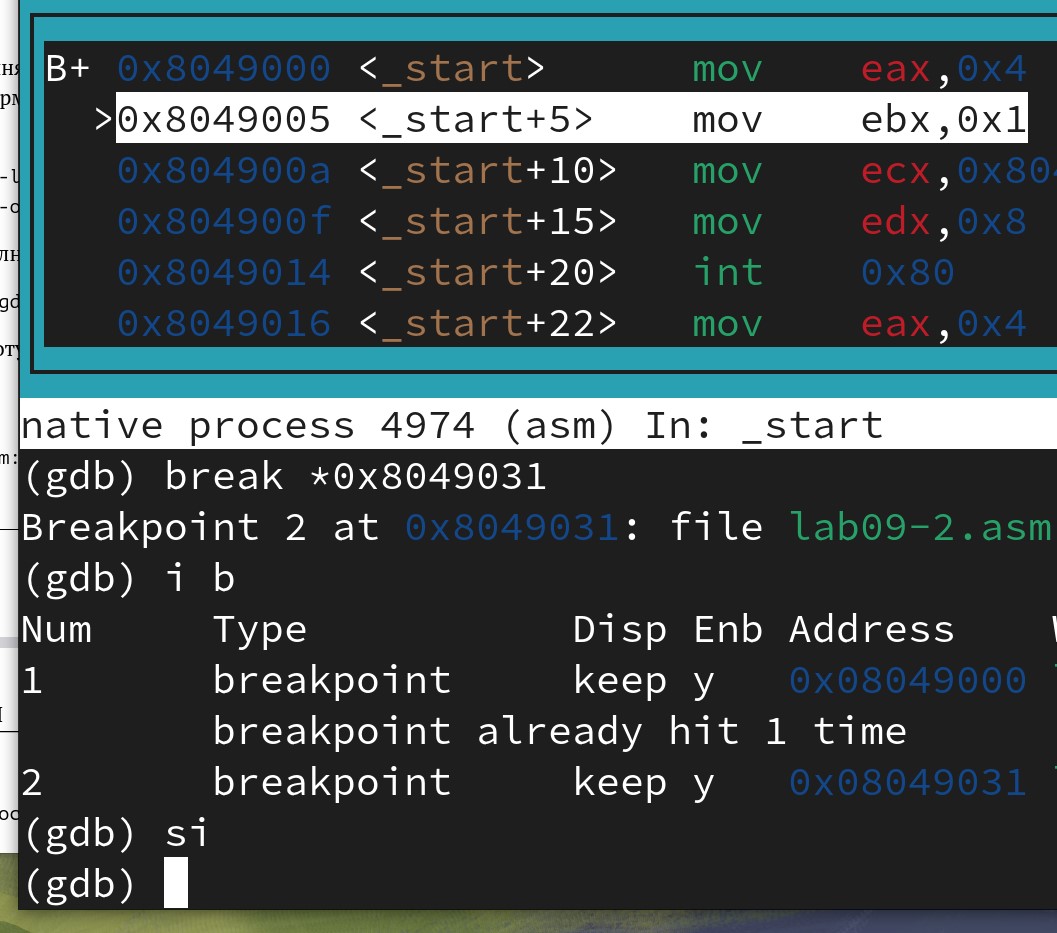


Рис. 15: Первое si

1. Второе si (Рис. 16).

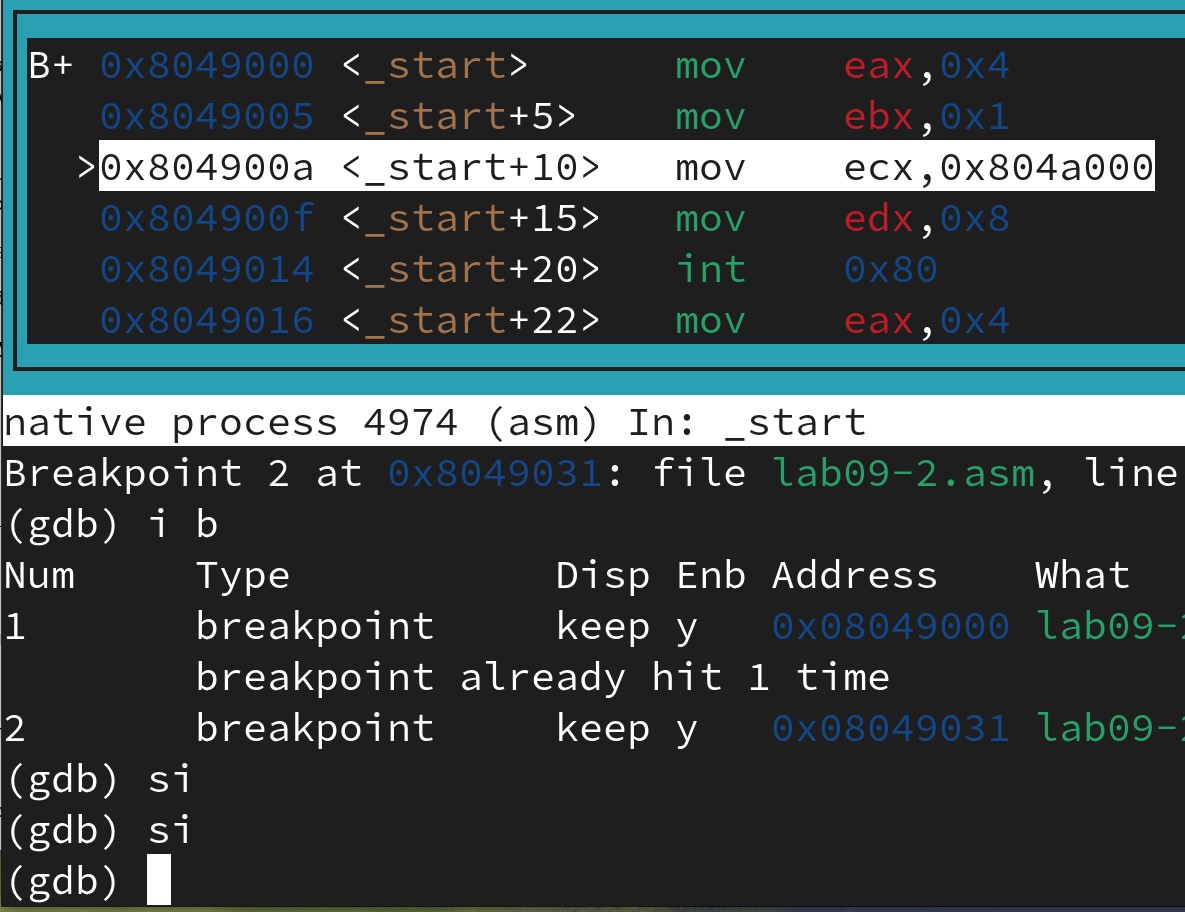


Рис. 16: Второе si

1. Третье si (Рис. 17).

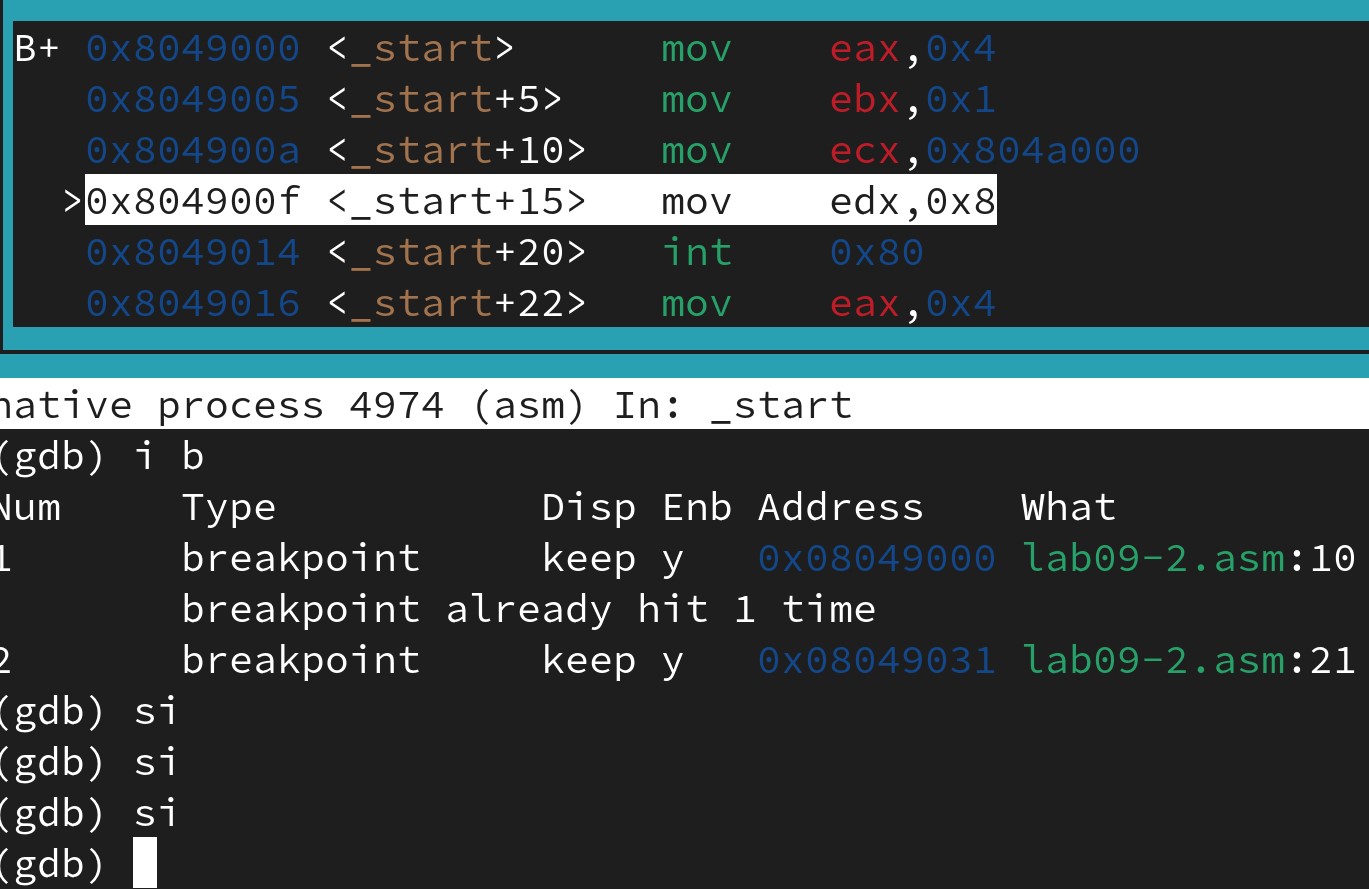


Рис. 17: Третье si

1. Четвертое si (рис. 18).

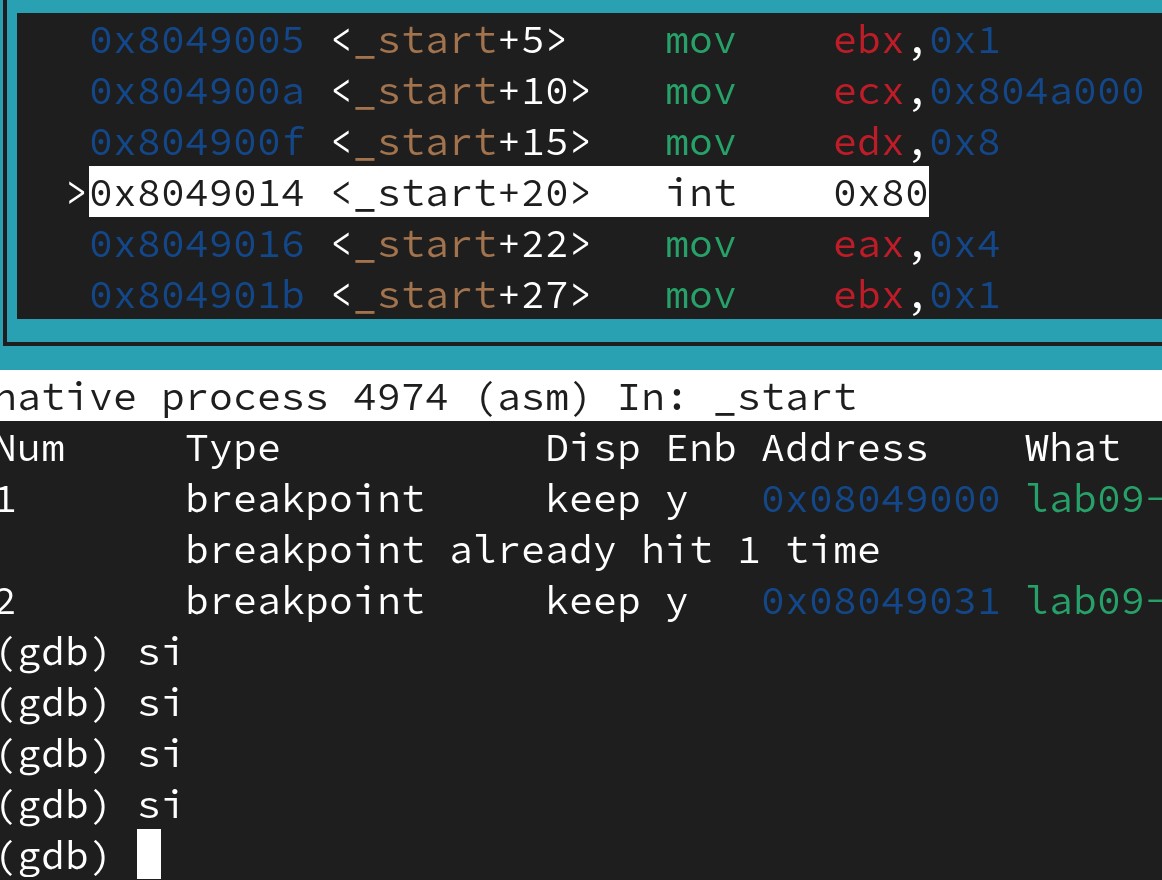


Рис. 18: Четвертое si

1. Пятое si (рис. 19).

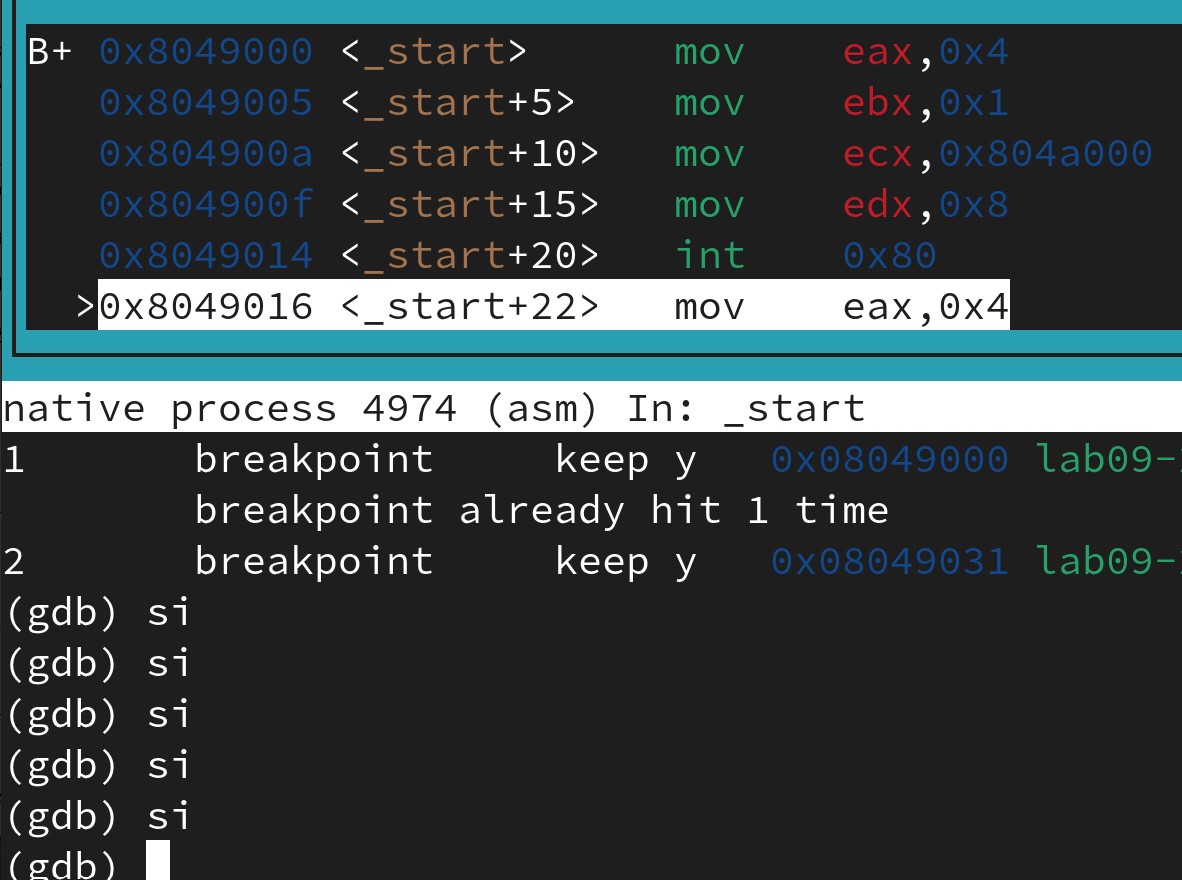


Рис. 19: Пятое si

1. Просматриваем значение всех регистров (Рис. 20)

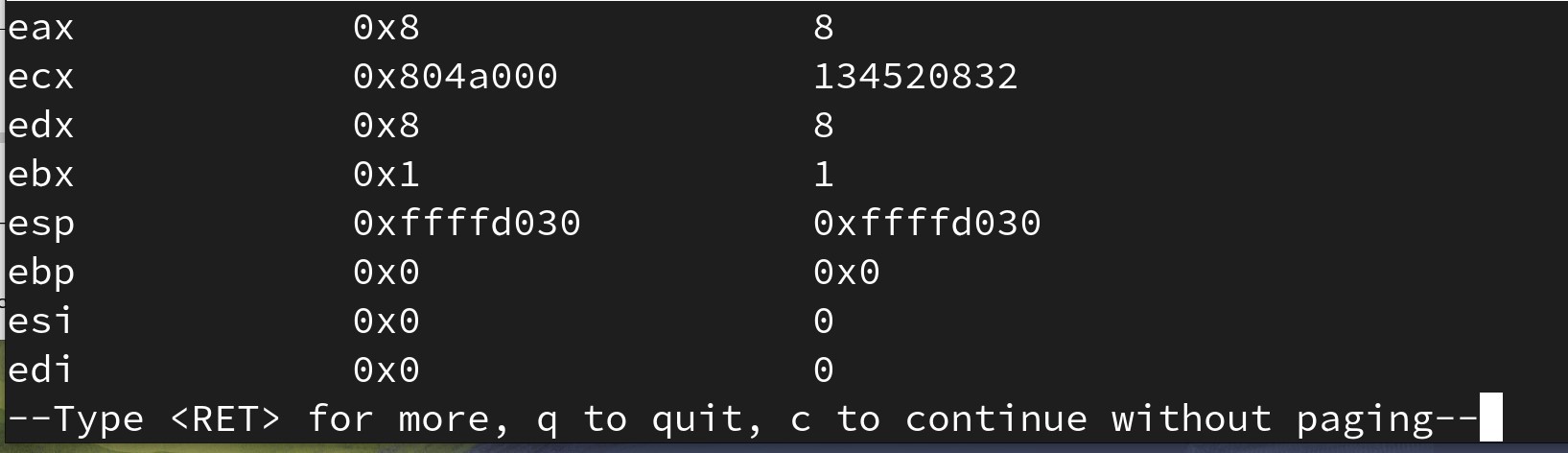


Рис. 20: Просмотр значений регистров

1. Смотрим значение msg1, а msg2 по адресу (Рис. 21)

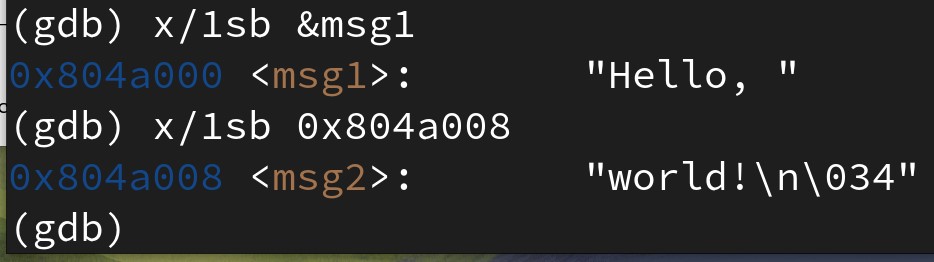


Рис. 21: Смотрим значения переменных

1. Изменяем два символа в предыдущих переменных (Рис. 22)

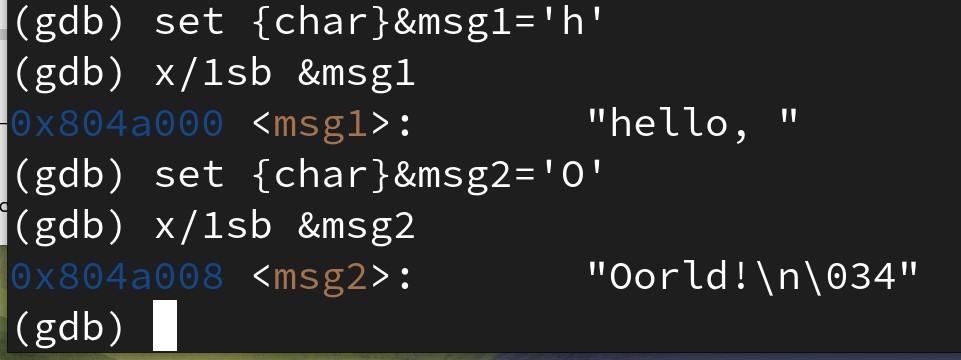


Рис. 22: Изменения значения регистра

1. Выводим значение edx в разных форматах (Рис. 23)

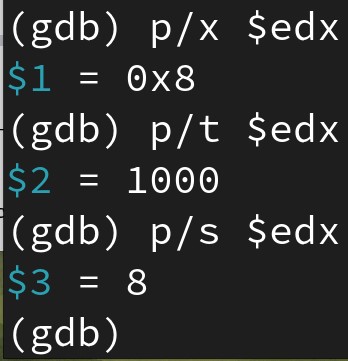


Рис. 23: Вывод значения

1. Меняем значение edx (Рис. 24)

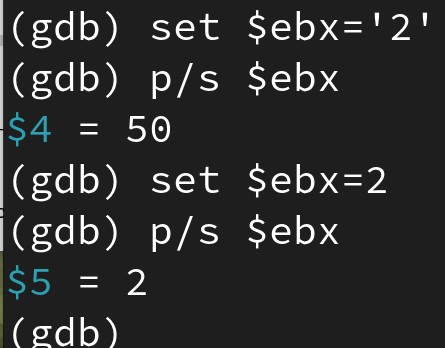


Рис. 24: Меняем значение

В первом случае выводится номер символа, во втором число.

Завершаем работу программы.

1. Копируем 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. Создаем исполняемый файл и загружаем его в отладчик с аргументами (Рис. 25)

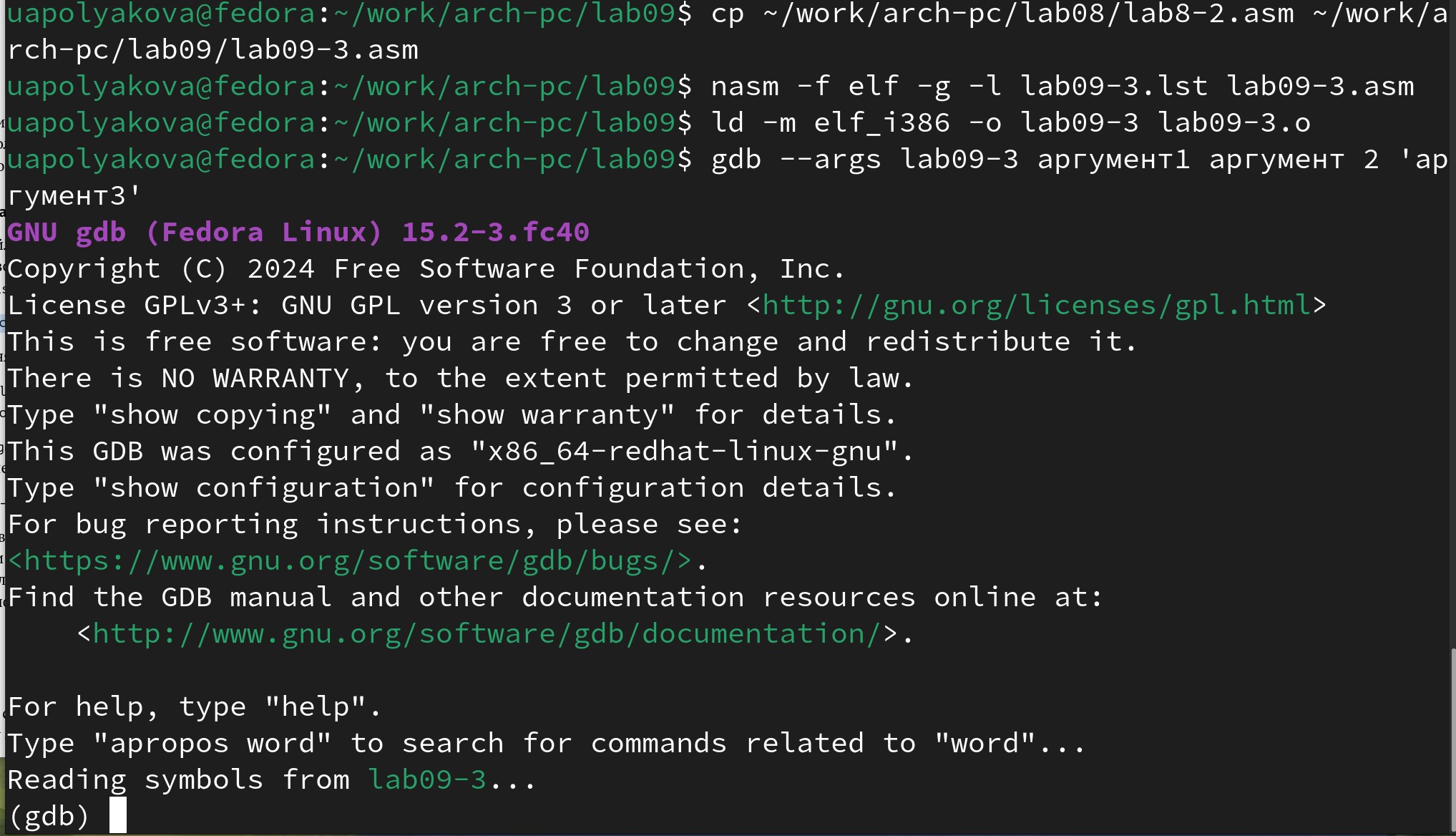


Рис. 25: Отладка с аргументами программы из лаб. 8

1. Ставим брейкпоинт и запускаем (Рис. 26)

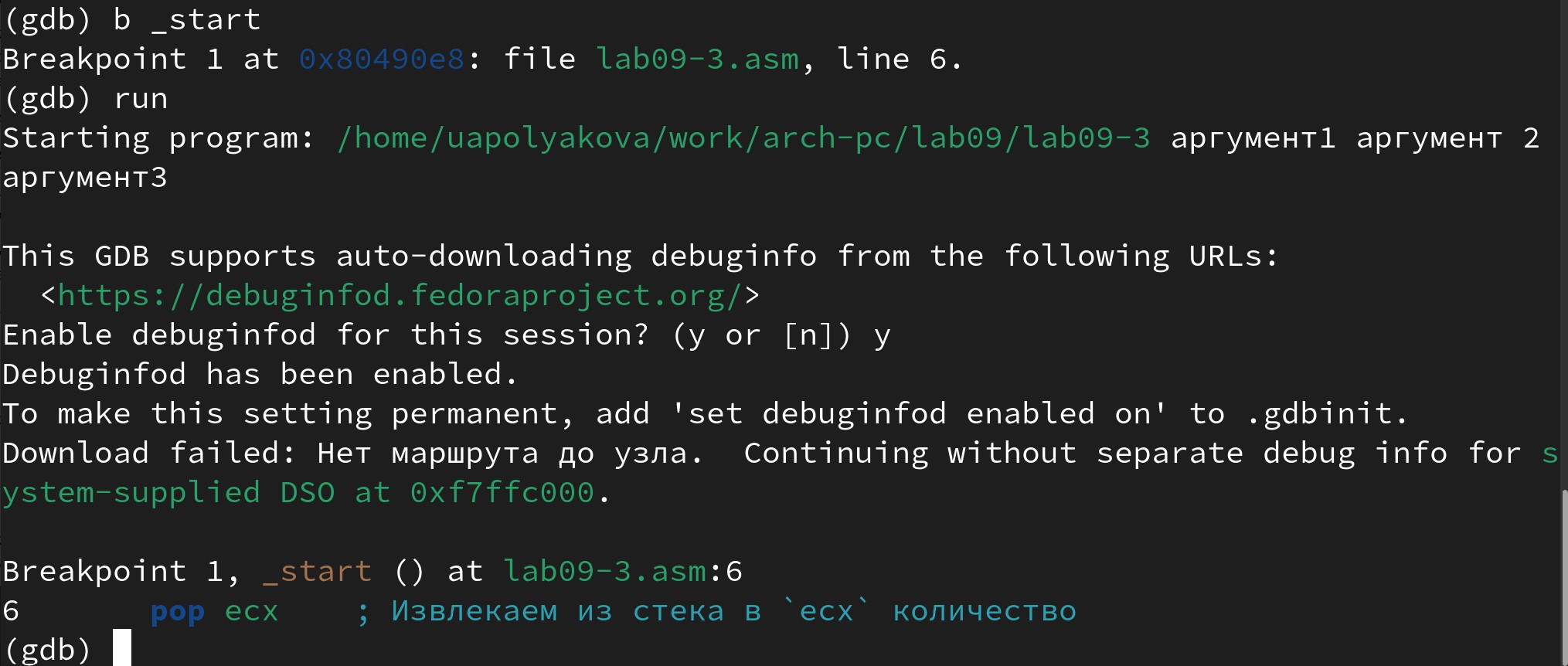


Рис. 26: Брейкпоинт и запуск

1. Смотрим значения из стека (Рис. 27)

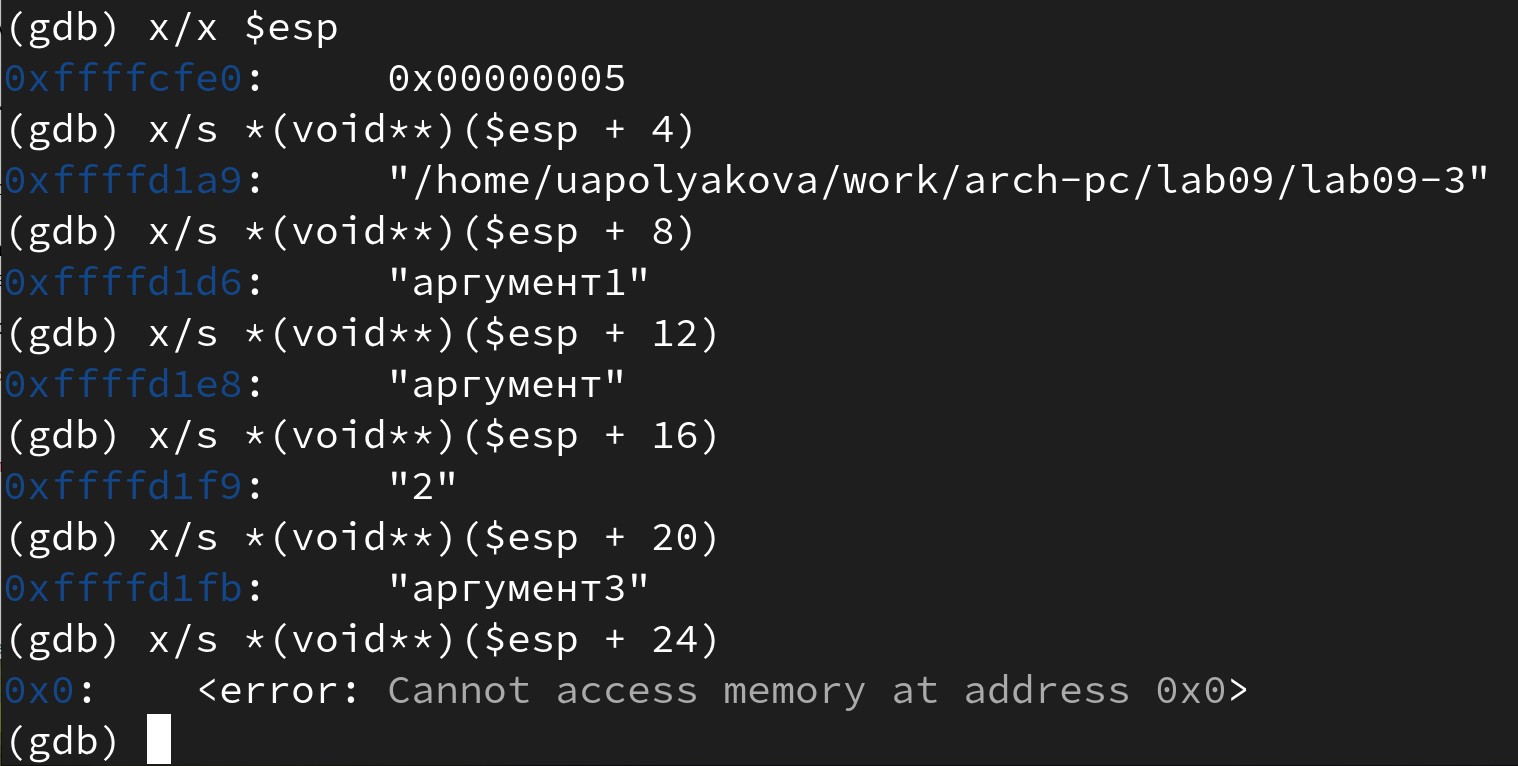


Рис. 27: Значения из стека

Шаг равен 4, чтобы учесть размер операнда.

# 3 Результаты выполнения заданий для самостоятельной работы

1. Преобразуйте программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции 𝑓(𝑥) как подпрограмму.

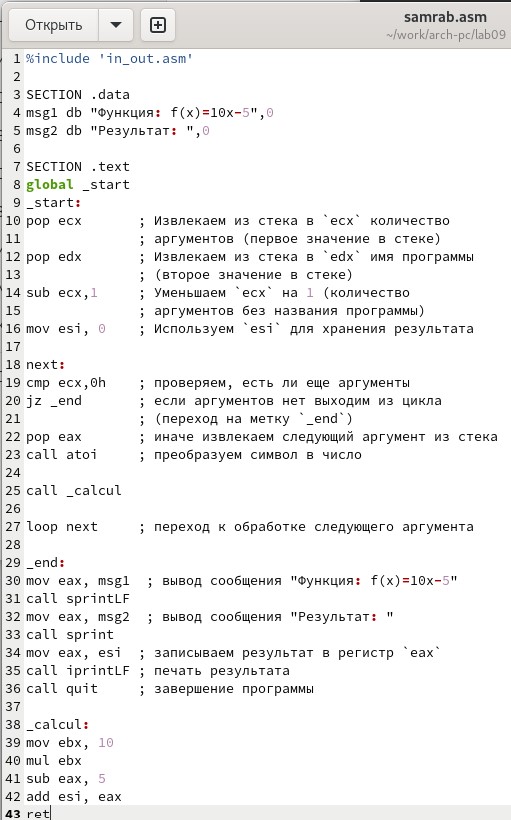


Рис. 28: Программа с вычислением в \_calc

1. Создаем исполняемый файл и запускаем.

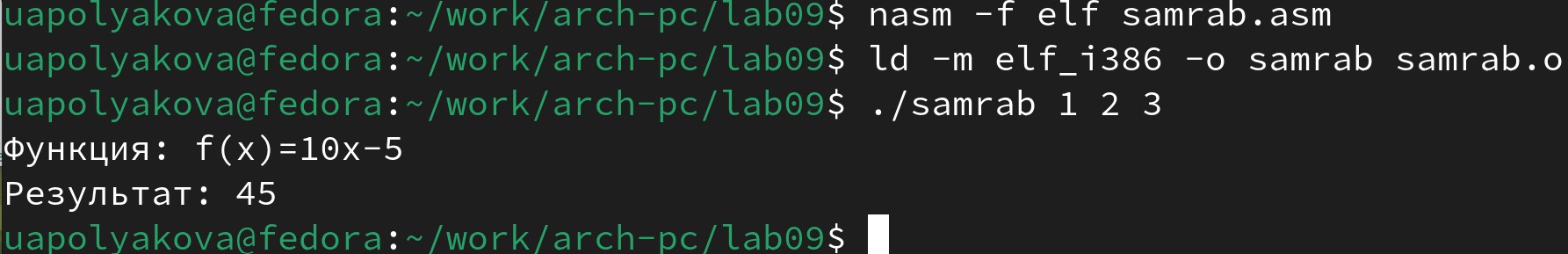


Рис. 29: Запуск программы

1. Проводим отладку данной в условии программы.

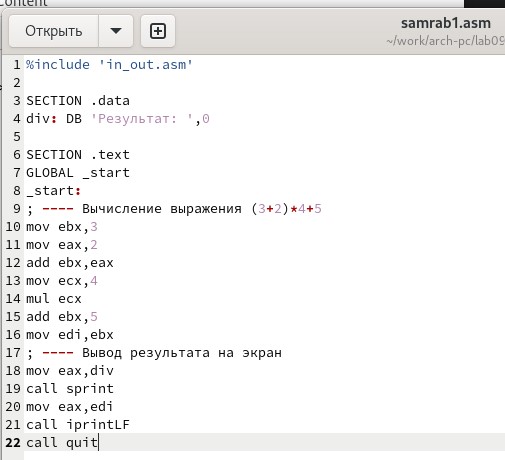


Рис. 30: Данная программа

Ошибка в том, что mul увеличивает eax на ecx, и этот результат никуда не записывается. Мы все время работаем с ebx и получается 3 + 2 + 5 = 10.

# 4 Вывод

Были приобретены навыки написания программ с использованием подпрограмм. Были изучены методы отладки при помощи GDB и его основные возможности.