О. Лабораторная работа №9. Понятие подпрограммы. Отладчик GDB

Архитектура компьютеров

Акрур Имад НКАбд-06-24

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 Описание результатов выполнения лабораторной работы:

## 2.1 Выполнение лабораторной работы

Я создал каталог для выполнения лабораторной работы № 9 и перешел в него. Затем я создал файл lab9-1.asm.

В качестве примера рассмотрим программу вычисления арифметического выражения с помощью подпрограммы calcul. В данном примере вводится с клавиатуры, а само выражение вычисляется в подпрограмме.(рис. 1) (рис. 2)

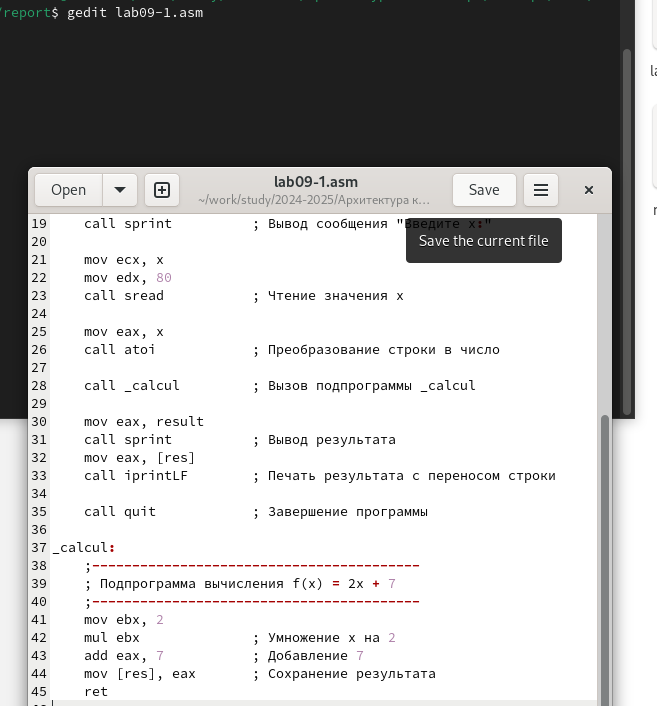


Рис. 1: Программа в файле lab9-1.asm

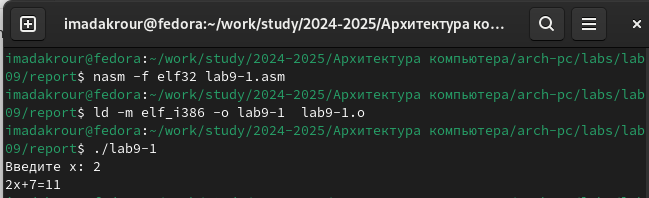


Рис. 2: Запуск программы lab9-1.asm

Изменил текст программы, добавив подпрограмму subcalcul в подпрограмму calcul, для вычисления выражения , где вводится с клавиатуры, . (рис. 3) (рис. 4)

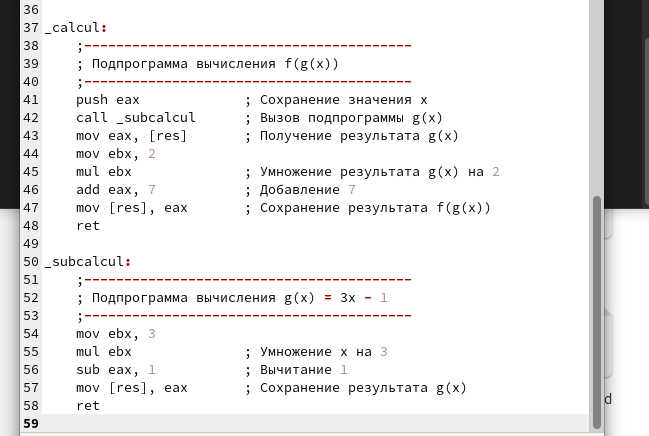


Рис. 3: Программа в файле lab9-1.asm

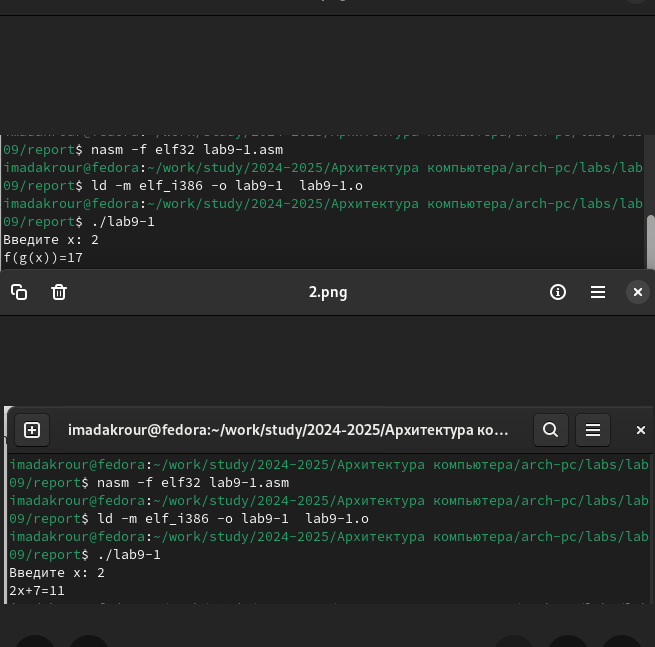


Рис. 4: Запуск программы lab9-1.asm

Создал файл lab9-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!). (рис. 5)



Рис. 5: Программа в файле lab9-2.asm

Получил исполняемый файл и добавил отладочную информацию с помощью ключа ‘-g’ для работы с GDB.

Загрузил исполняемый файл в отладчик GDB и проверил работу программы, запустив ее с помощью команды ‘run’ (сокращенно ‘r’). (рис. 6)

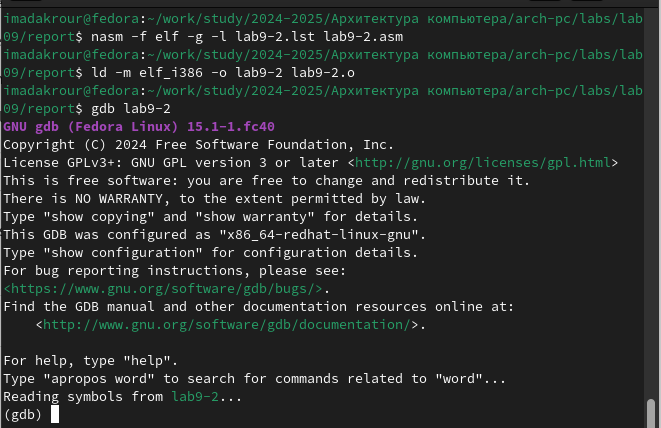


Рис. 6: Запуск программы lab9-2.asm в отладчике

Для более подробного анализа программы, установил точку остановки на метке ‘start’, с которой начинается выполнение любой ассемблерной программы, и запустил ее. Затем просмотрел дизассемблированный код программы.(рис. 7) (рис. 8)

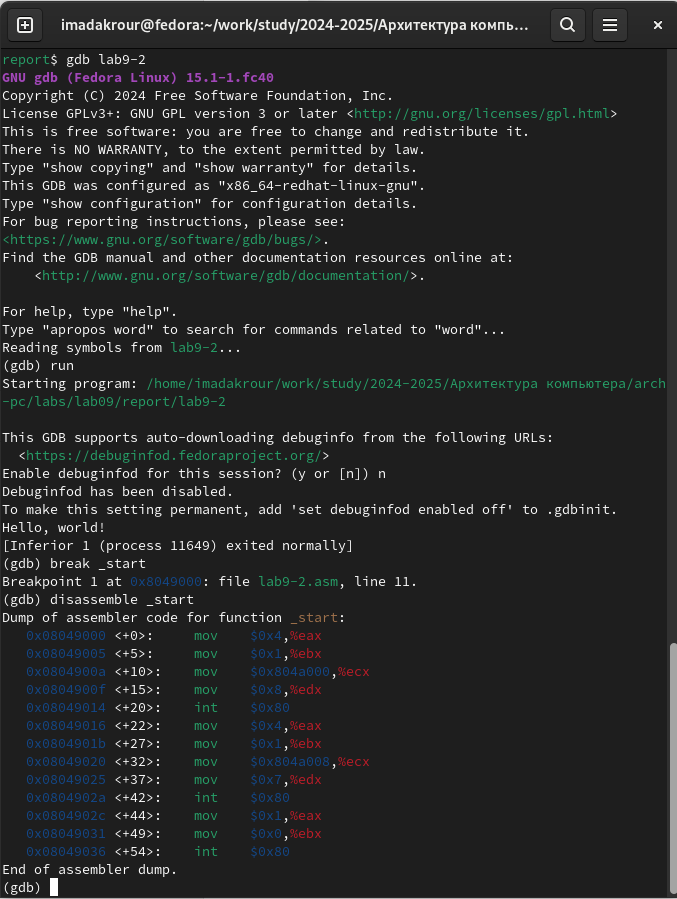


Рис. 7: Дизассемблированный код

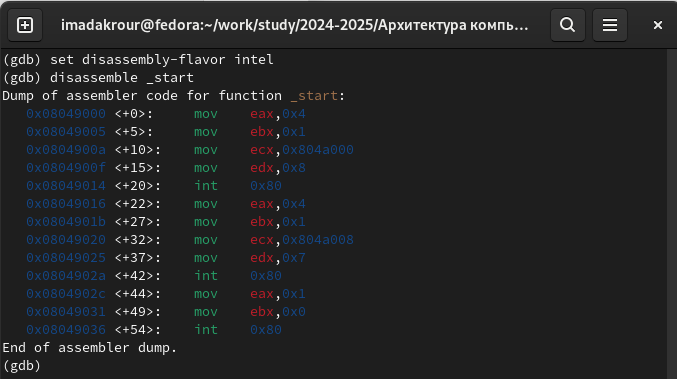


Рис. 8: Дизассемблированный код в режиме интел

Для проверки точки остановки по имени метки ’\_start’, использовал команду ‘info breakpoints’ (сокращенно ‘i b’). Затем установил еще одну точку остановки по адресу инструкции, определив адрес предпоследней инструкции ‘mov ebx, 0x0’. (рис. 9)

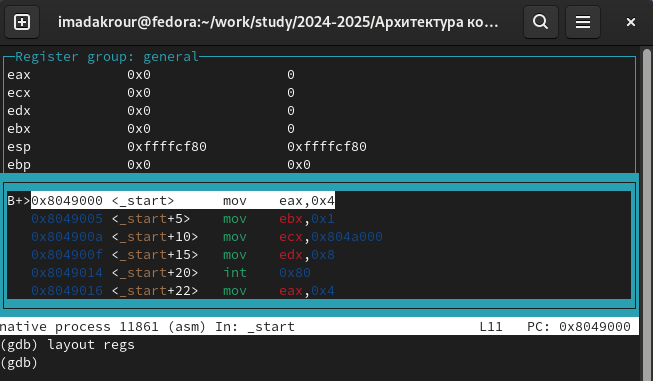


Рис. 9: Точка остановки

В отладчике GDB можно просматривать содержимое ячеек памяти и регистров, а также изменять значения регистров и переменных. Выполнил 5 инструкций с помощью команды ‘stepi’ (сокращенно ‘si’) и отследил изменение значений регистров. (рис. 10) (рис. 11)

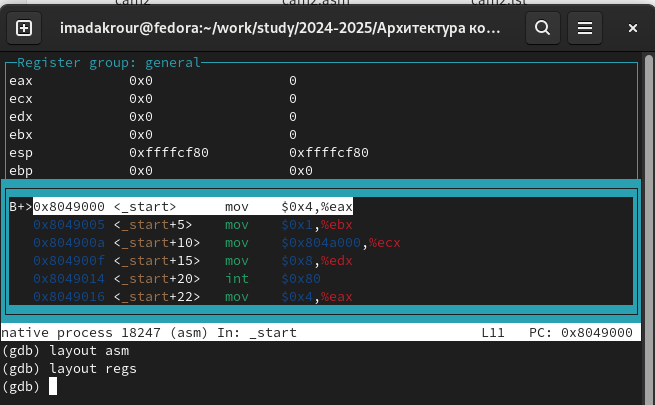


Рис. 10: Изменение регистров

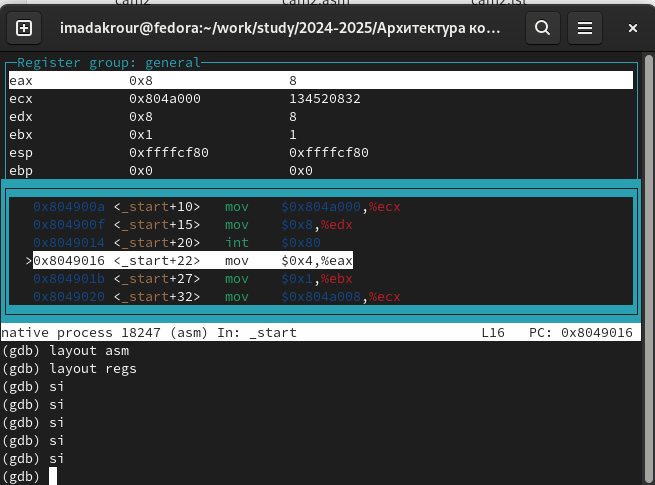


Рис. 11: Изменение регистров

Просмотрел значение переменной msg1 по имени и получил нужные данные.

Просмотрел значение переменной msg1 по имени и получил нужные данные.

Для изменения значения регистра или ячейки памяти использовал команду set, указав имя регистра или адрес в качестве аргумента. Изменил первый символ переменной msg1. (рис. 12)

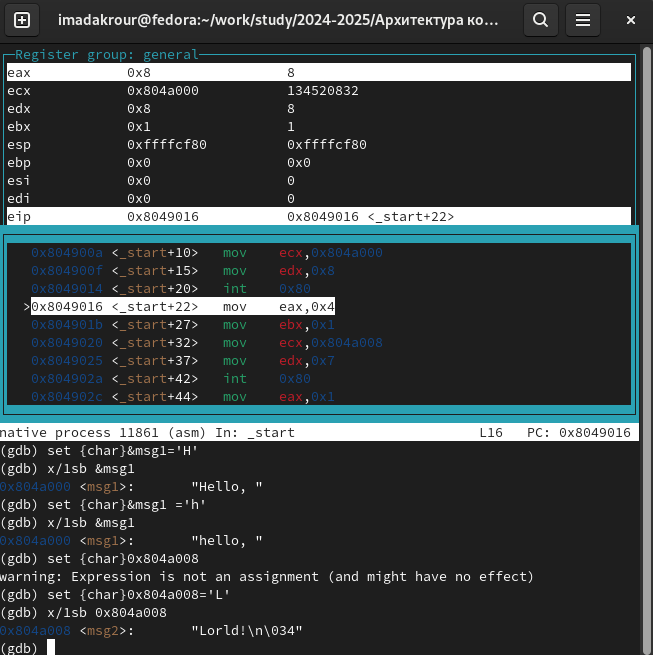


Рис. 12: Изменение значения переменной

Для изменения значения регистра или ячейки памяти использовал команду set, указав имя регистра или адрес в качестве аргумента. Изменил первый символ переменной msg1.(рис. 13)

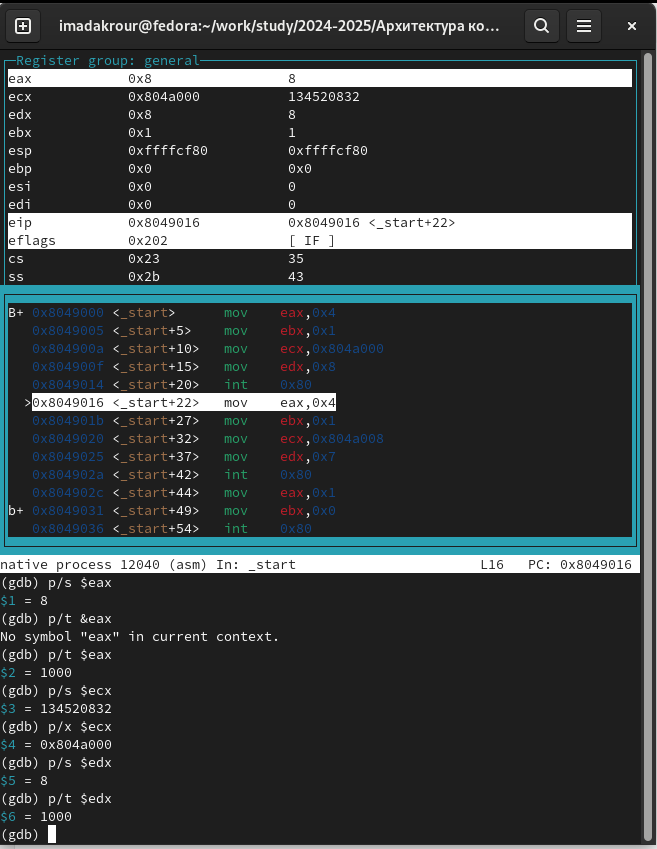


Рис. 13: Вывод значения регистра

С помощью команды set изменил значение регистра ebx на нужное значение. (рис. 14)

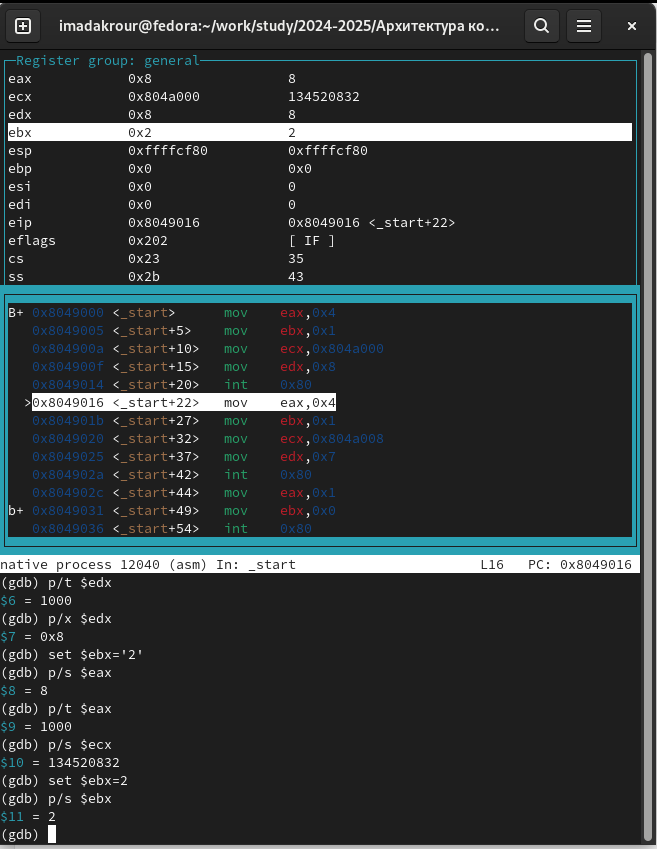


Рис. 14: Вывод значения регистра

Скопировал файл lab8-2.asm, созданный во время выполнения лабораторной работы №8, который содержит программу для вывода аргументов командной строки. Создал исполняемый файл из скопированного файла.

Для загрузки программы с аргументами в gdb использовал ключ –args и загрузил исполняемый файл в отладчик с указанными аргументами.

Установил точку останова перед первой инструкцией программы и запустил ее.

Адрес вершины стека, содержащий количество аргументов командной строки (включая имя программы), хранится в регистре esp. По этому адресу находится число, указывающее количество аргументов. В данном случае видно, что количество аргументов равно 5, включая имя программы lab9-3 и сами аргументы: аргумент1, аргумент2 и ‘аргумент 3’.

Просмотрел остальные позиции стека. По адресу [esp+4] находится адрес в памяти, где располагается имя программы. По адресу [esp+8] хранится адрес первого аргумента, по адресу [esp+12] - второго и так далее. (рис. 15)

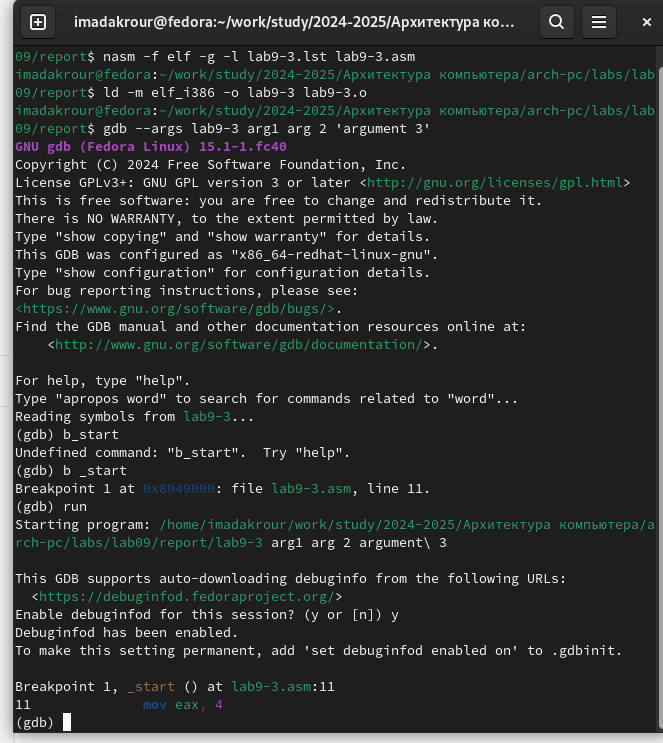


Рис. 15: Вывод значения регистра

Шаг изменения адреса равен 4, так как каждый следующий адрес на стеке находится на расстоянии 4 байт от предыдущего ([esp+4], [esp+8], [esp+12]).

# 3 Описание результатов выполнения заданий для самостоятельной работы:

## 3.1 Oписание выполняемого задания

Преобразовал программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции f(x) как подпрограмму. (рис. 16) (рис. 17)

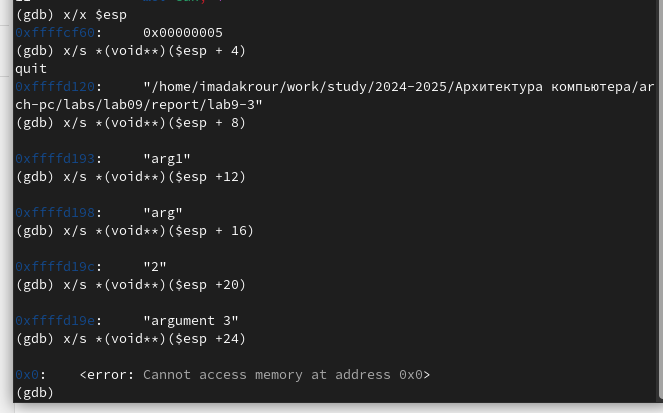


Рис. 16: Программа в файле prog-1.asm

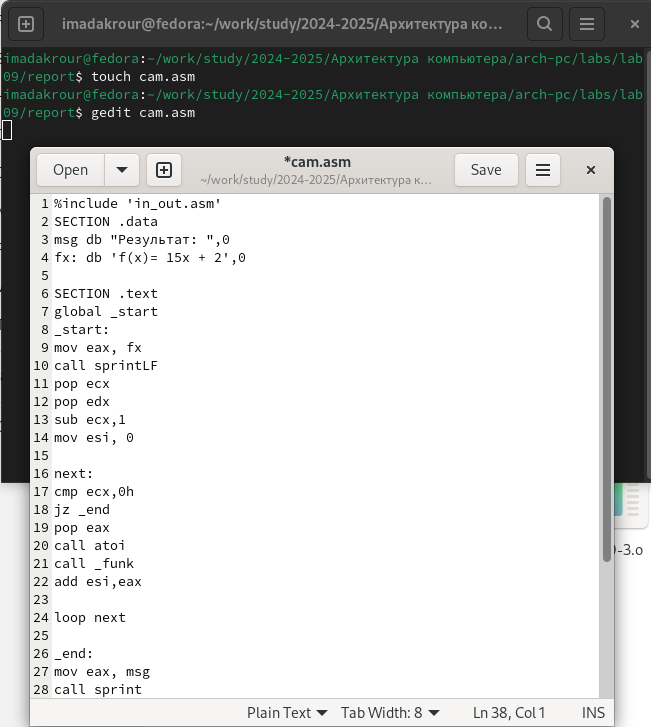


Рис. 17: Запуск программы prog-1.asm

В листинге приведена программа вычисления выражения . При запуске данная программа дает неверный результат. Проверил это, анализируя изменения значений регистров с помощью отладчика GDB.

Определил ошибку - перепутан порядок аргументов у инструкции add. Также обнаружил, что по окончании работы в edi отправляется ebx вместо eax.(рис. 18)

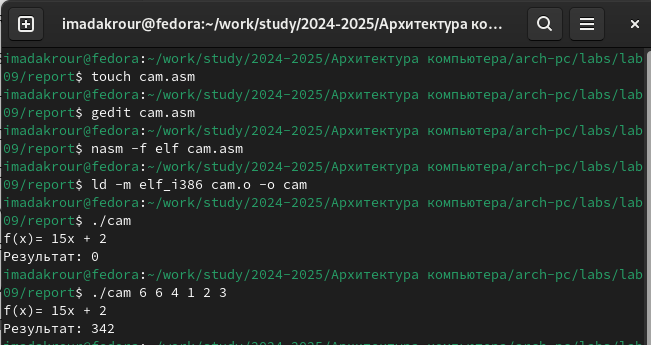


Рис. 18: Код с ошибкой

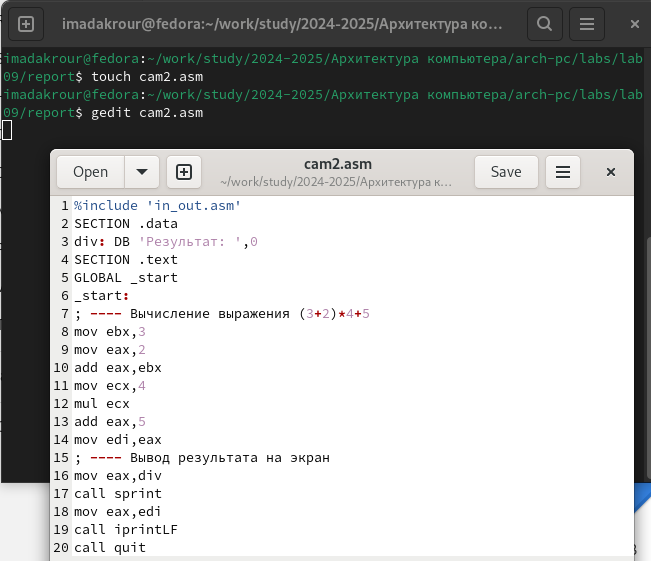


Рис. 19: Отладка

Отмечу, что перепутан порядок аргументов у инструкции add и что по окончании работы в edi отправляется ebx вместо eax (рис. 19)

Исправленный код программы (рис. 20) (рис. 21)

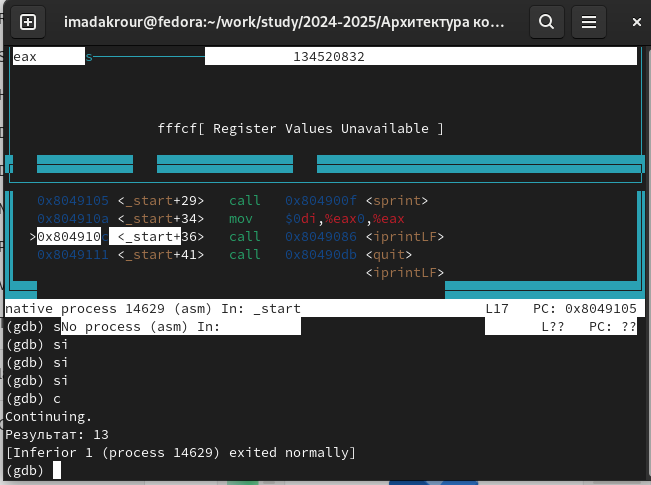


Рис. 20: Код исправлен

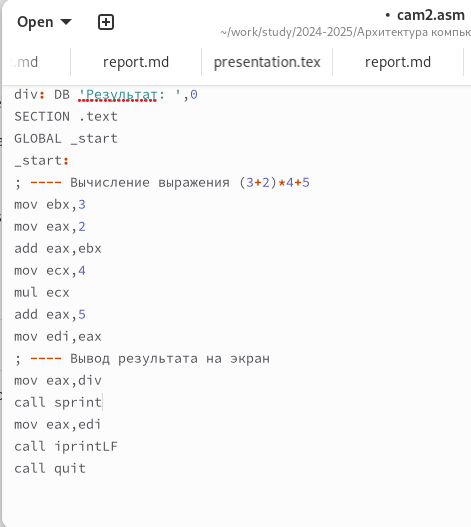


Рис. 21: Проверка работы

# 4 Выводы

Освоили работy с подпрограммами и отладчиком.