Лабораторная работа №9.

Понятие подпрограммы.Отладчик GDB.

Швед Карина Дмитриевна

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

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

# 2 Ход работы

Я создала каталог для выполнения лабораторной работы № 9, перейшла в него и со- здала файл lab09-1.asm

В качестве примера рассмотрим программу вычисления арифметического выражения 𝑓(𝑥) = 2𝑥 + 7 с помощью подпрограммы \_calcul. В данном примере 𝑥 вводится с клавиатуры, а само выражение вычисляется в подпрограмме. Я внимательно изучила текст программы (Листинг 9.1).Ввела в файл lab09-1.asm текст программы из листинга 9.1. Создала исполняемый файл (рис. 1) и проверила его работу (рис. 2).

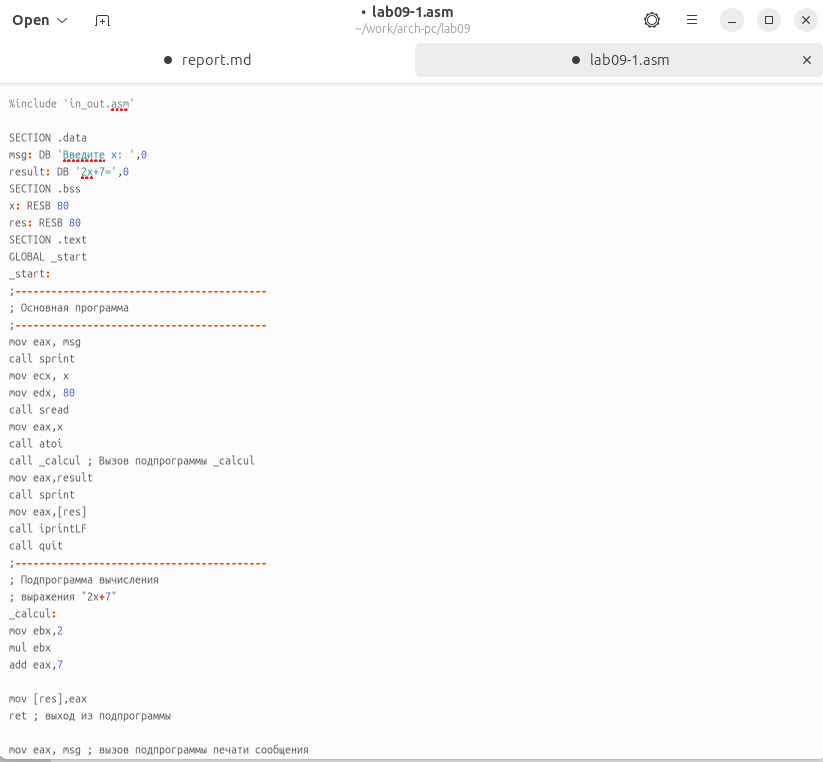


Рис. 1: код программы lab09-1.asm

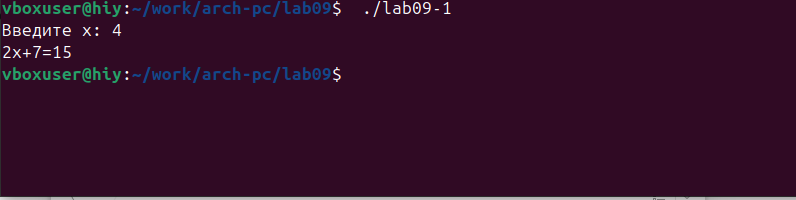


Рис. 2: работа программы lab09-1.asm

Я изменила 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. Я создала исполняемый файл (рис. 3) и проверила его работу (рис. 4)

|  |
| --- |
| Рис. 3: код программы lab09-1.asm |

Рис. 3: код программы lab09-1.asm

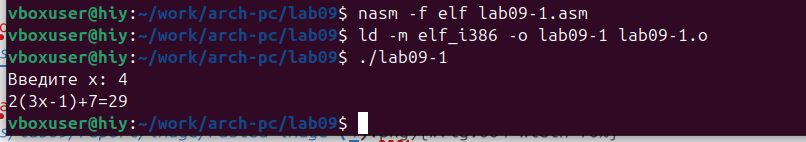


Рис. 4: работа программы lab09-1.asm

Я создала файл lab09-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!)

Получила исполняемый файл и добавила отладочную информацию с помощью ключа ‘-g’ для работы с GDB.(рис. 5)

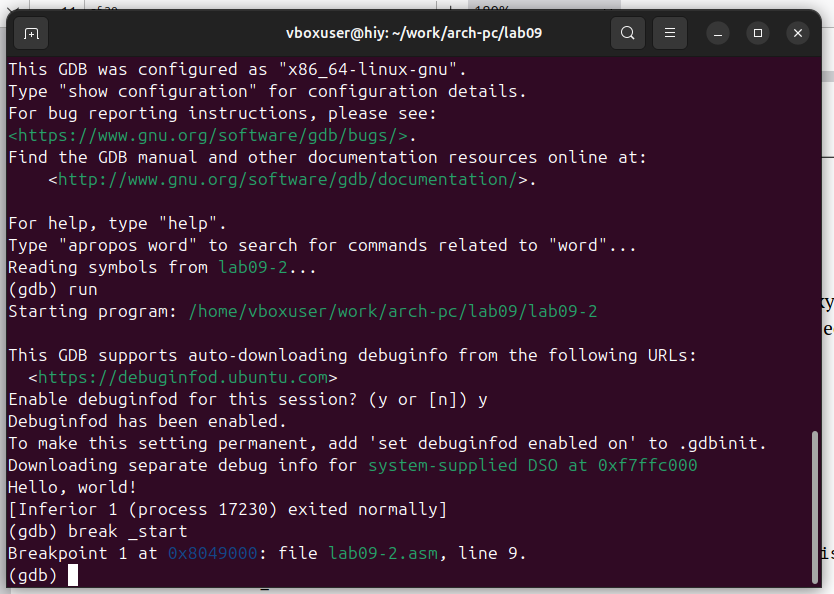


Рис. 5: запуск программы lab09-2.asm в отладчике

Для более подробного анализа программы, установила точку остановки на метке ‘start’, с которой начинается выполнение любой ассемблерной программы, и запустила ее. Затем просмотрела дизассемблированный код программы (рис. 6) (рис. 7)

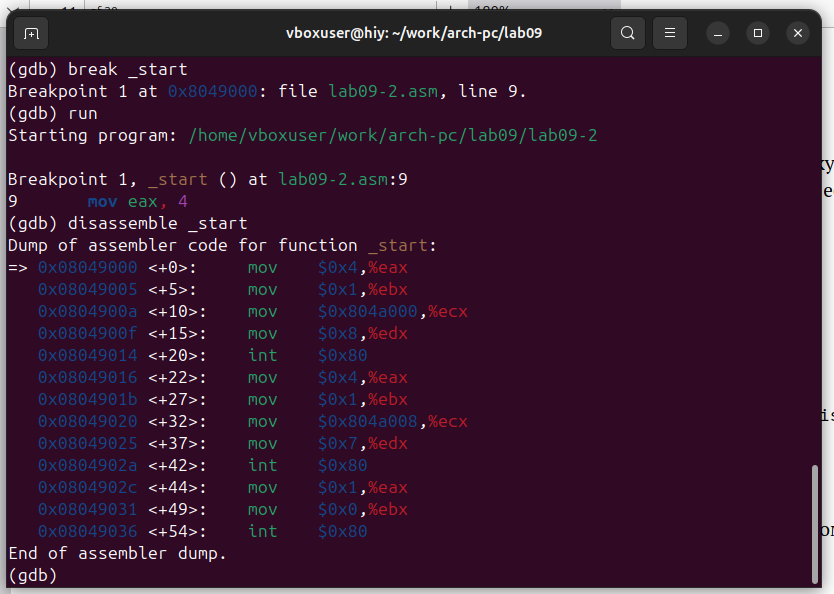


Рис. 6: Дизассемблированный код

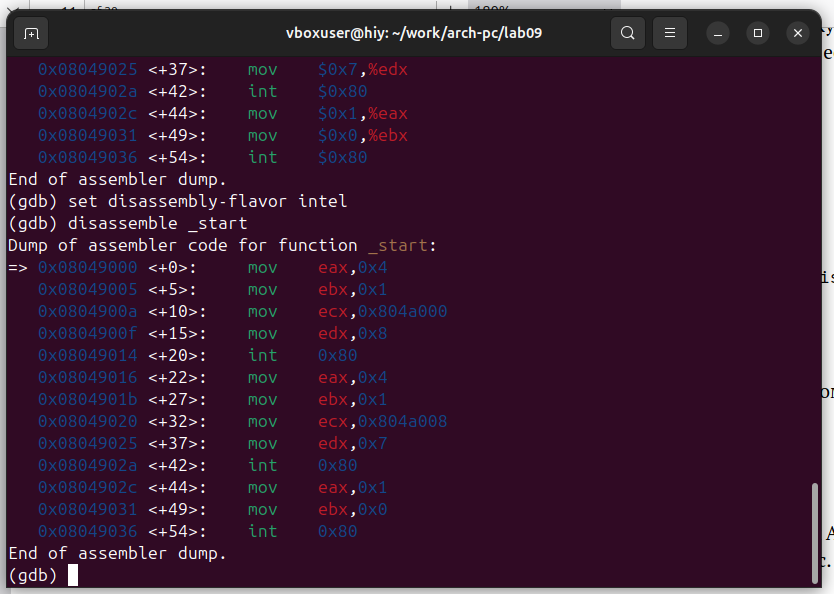


Рис. 7: Дизассемблированный код в режиме intel

Включила режим псевдографики для более удобного анализа программы Для проверки точки остановки по имени метки ’\_start’, использовала команду ‘info breakpoints’ (сокращенно ‘i b’). Затем установила еще одну точку остановки по адресу инструкции, определив адрес предпоследней инструкции ‘mov ebx, 0x0’. (рис. 7)

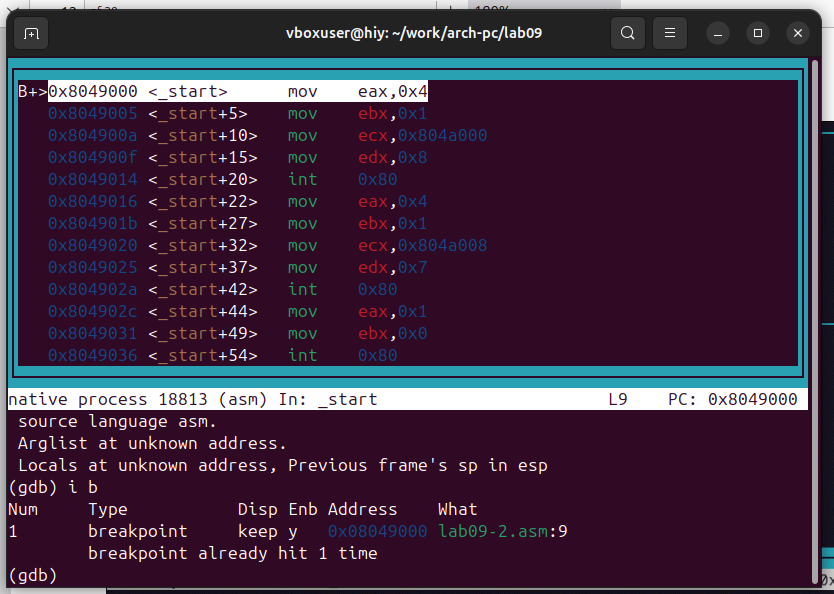


Рис. 8: точка останова

Установила еще одну точку останова по адресу инструкции mov ebx,0x0 (рис. 9)

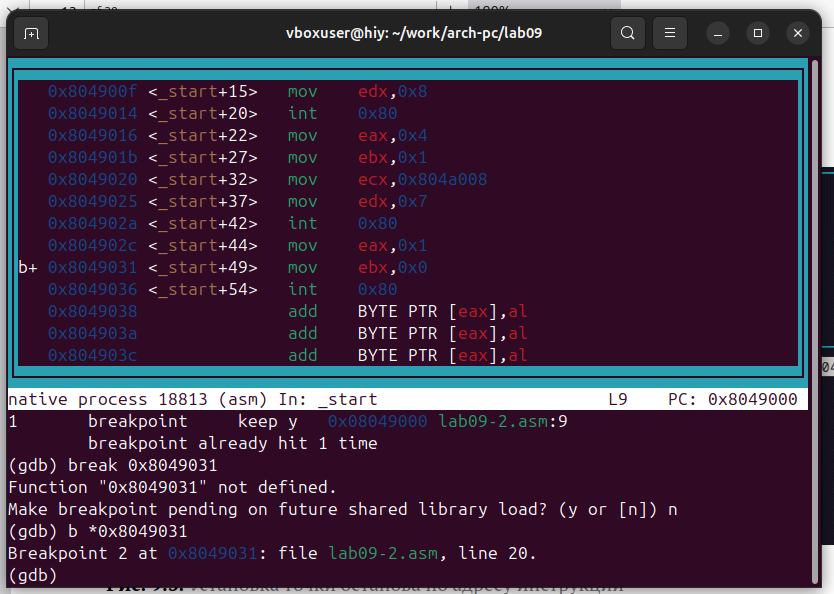


Рис. 9: точка останова

Посмотрите информацию о всех установленных точках останова (рис. 10)

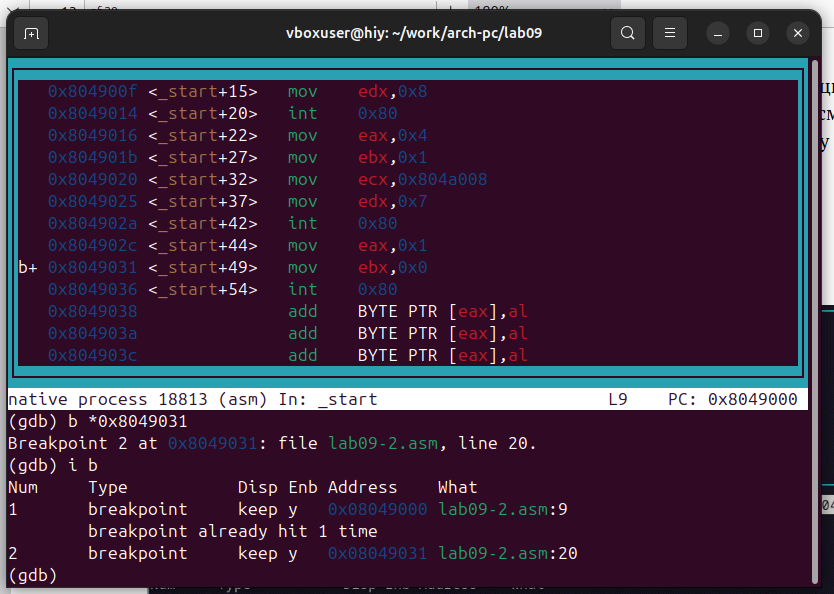


Рис. 10: все точки останова

Отладчик может показывать содержимое ячеек памяти и регистров, а при необходимости позволяет вручную изменять значения регистров и переменных.Выполнила 5 инструкций с помощью команды stepi (или si) и проследила за изменением значений регистров (рис. 11)

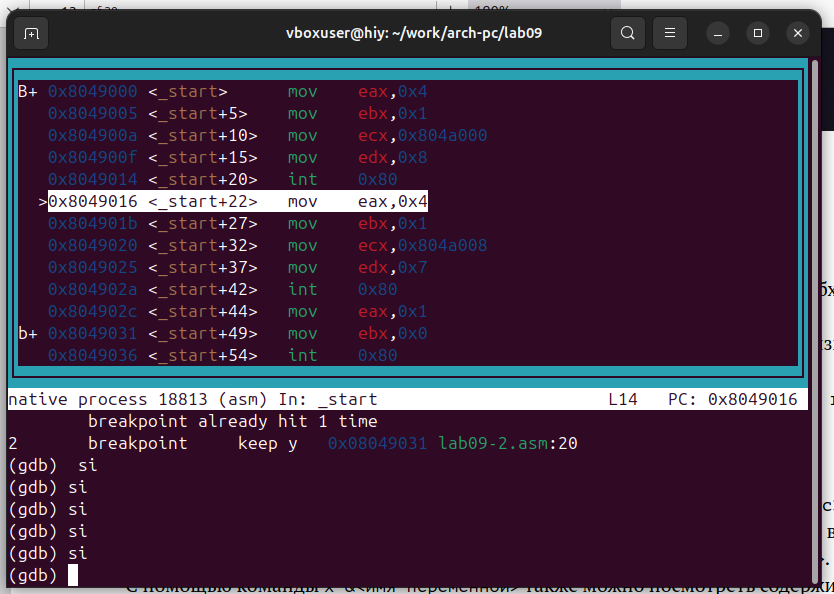


Рис. 11: изменение регистров

Просмотрела значение переменной msg1 по имени и получил нужные данные. Просмотрела значение переменной msg1 по имени и получил нужные данные. Для изменения значения регистра или ячейки памяти использовала команду set, указав имя регистра или адрес в качестве аргумента.Изменила первый символ переменной msg1 и первый символ переменной msg2 (рис. 12)

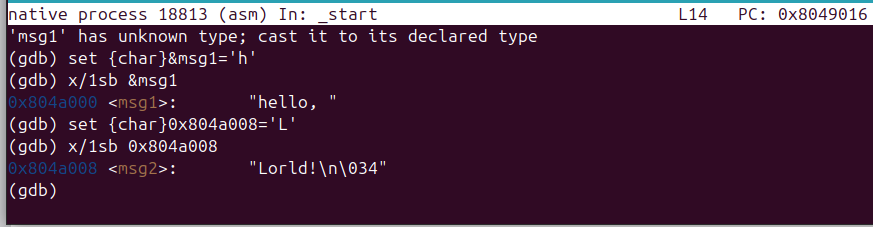


Рис. 12: изменение значения переменной

Вывела в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx. (рис. 13) С помощью команды set изменила значение регистра ebx (рис. 14)

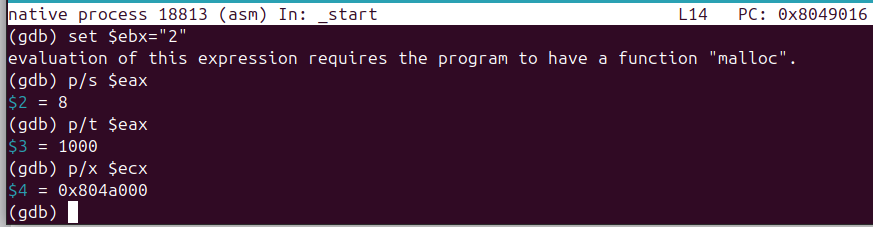


Рис. 13: вывод значения регистра

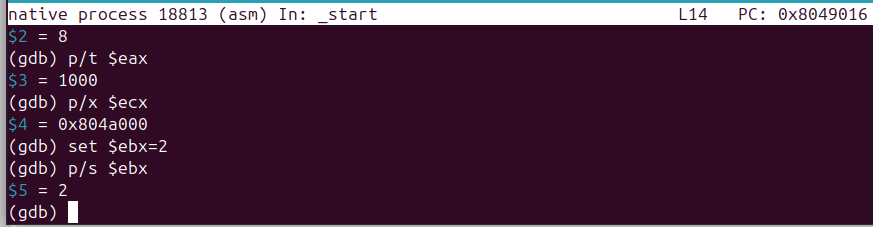


Рис. 14: вывод значения регистра

Скопировала файл lab8-2.asm, созданный во время выполнения лабораторной работы №8, который содержит программу для вывода аргументов командной строки. Создала исполняемый файл из скопированного файла. Для загрузки программы с аргументами в gdb использовала ключ –args и загрузил исполняемый файл в отладчик с указанными аргументами. Установила точку останова перед первой инструкцией программы и запустила 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’. Просмотрела 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)

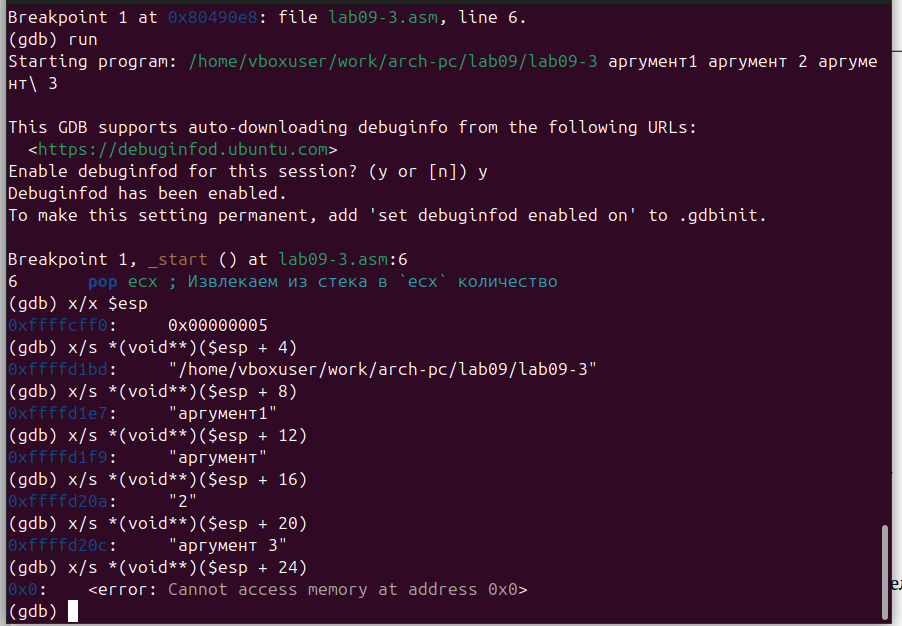


Рис. 15: вывод значения регистра

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 Самостоятельная работа

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

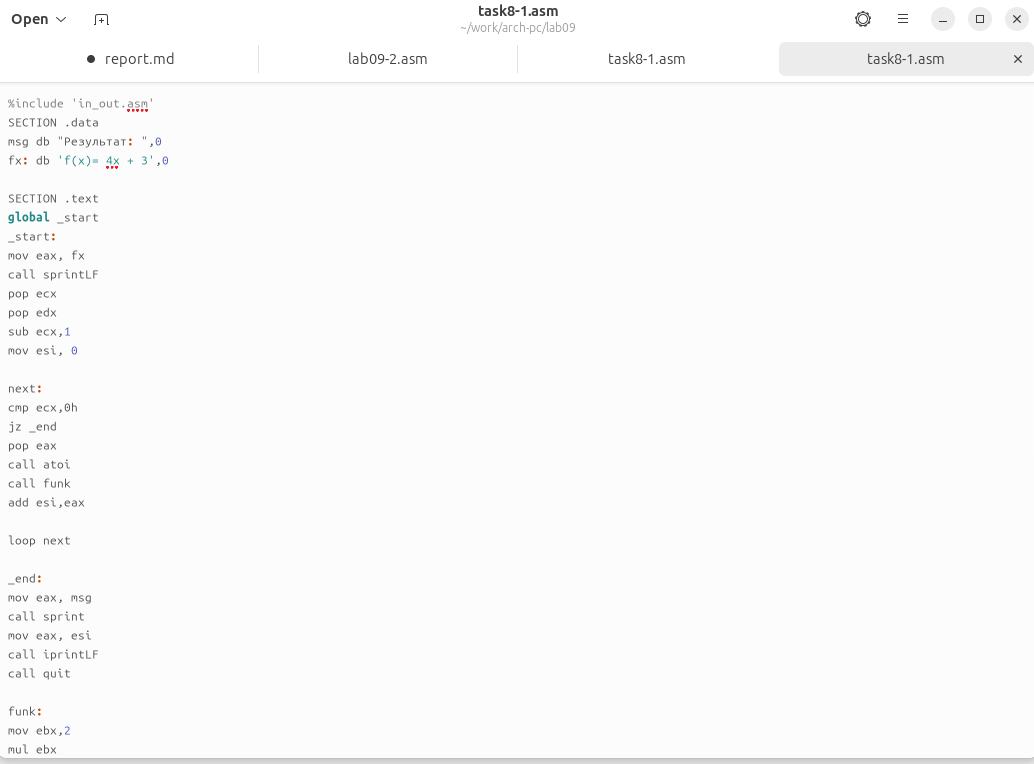


Рис. 16: программа в файле task8-1.asm

Создала исполняемый файл и проверила его работу (рис. 17)

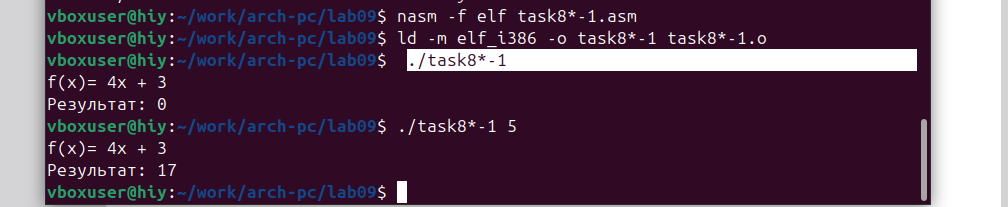


Рис. 17: программа в файле task8-1.asm

2)В листинге приведена программа вычисления выражения (3+2)\*4+5. При запуске данная программа дает неверный результат. Проверила это, анализируя изменения значений регистров с помощью отладчика GDB. Определила ошибку - перепутан порядок аргументов у инструкции add. Также обнаружила, что по окончании работы в edi отправляется ebx вместо eax (рис. 18)

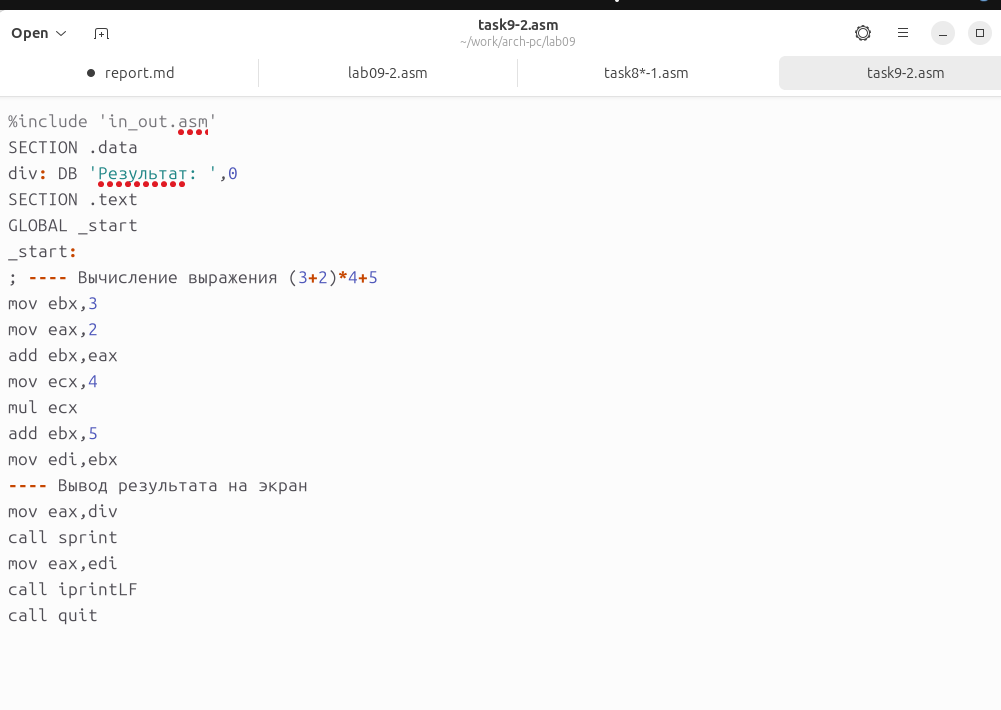


Рис. 18: код с ошибкой

перепутан порядок аргументов у инструкции add и что по окончании работы в edi отправляется ebx вместо eax

(рис. 19)

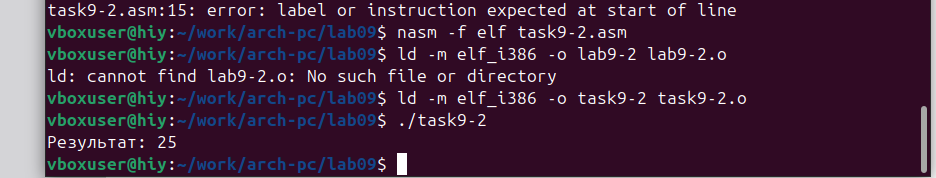


Рис. 19: код исправлен

# 4 Выводы

Я приобрела навыки написания программ с использованием подпрограмм, а также познакомилась с методами отладки при помощи GDB и его основными возможностями.