Устанавливаем компилятор GCC и все необходимые для него компоненты



Рис.1

Создаем файл с кодом для компиляции



Рис.2

Открываем его и вставляет туда код для теста

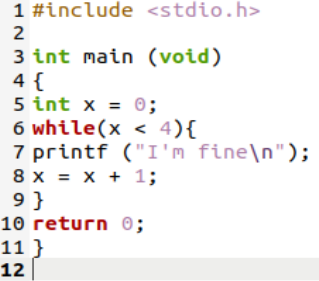


Рис.3

Компилируем



Рис.4

Открываем и проверяем наш файл

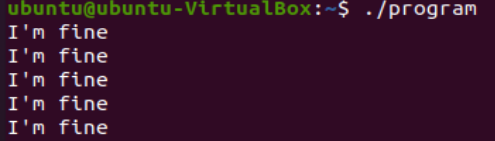


Рис.5

**Дизассемблирование**

Устанавливаем необходимые пакеты для работы с radare2



Рис. 6

Установка radare2:





Устанавливаем необходимые компоненты для граф.оболочки «Iaito»

Рис.6

Открываем папку и открываем файл



Рис.7



Рис.8

Устанавливаем графический интерфейс «iaito» (установка может выполняться из любого места)



Рис.9

Устанавливаем декомпилятор «r2ghidra» (установка может выполняться из любого места)



Рис.10

Запускаем «Iaito»

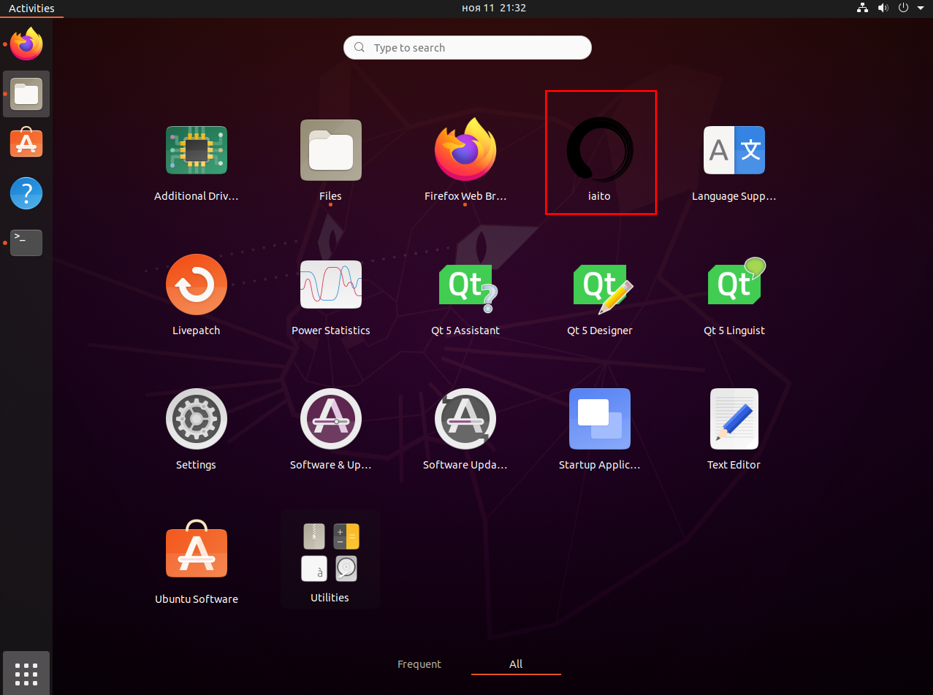


Рис.11

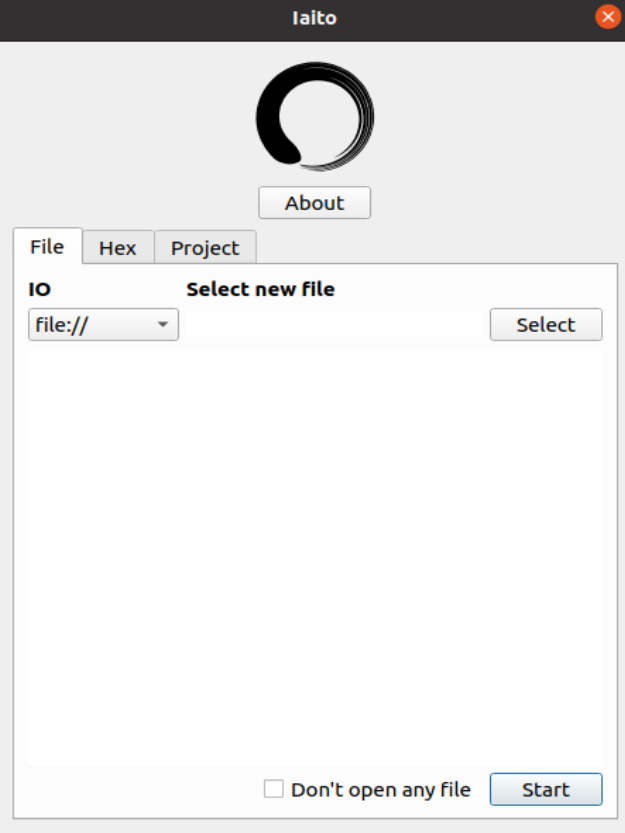


Рис.12

Запускаем и проверяем тестовый файл (после можно закрыть)

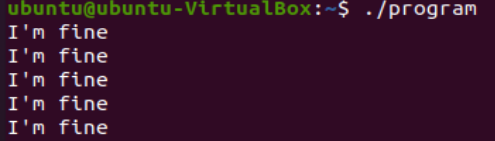


Рис.12

Открываем тестовый файл в «radare2». Настройки оставляем стандартные

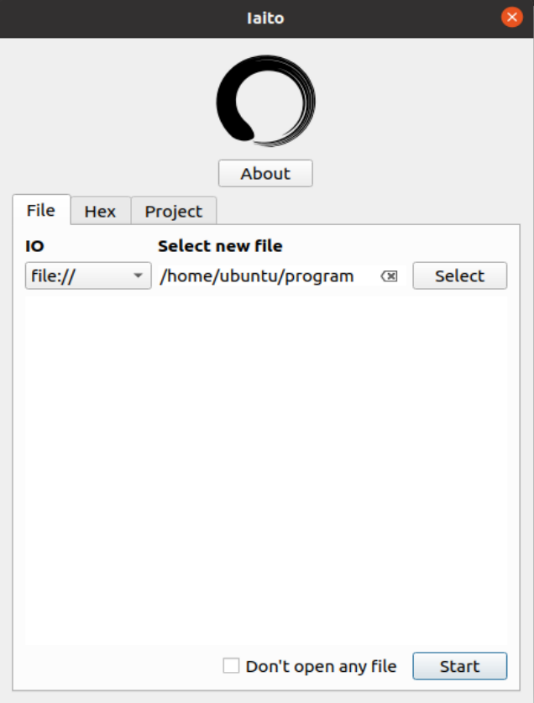


Рис.13

Открываем слева «main»

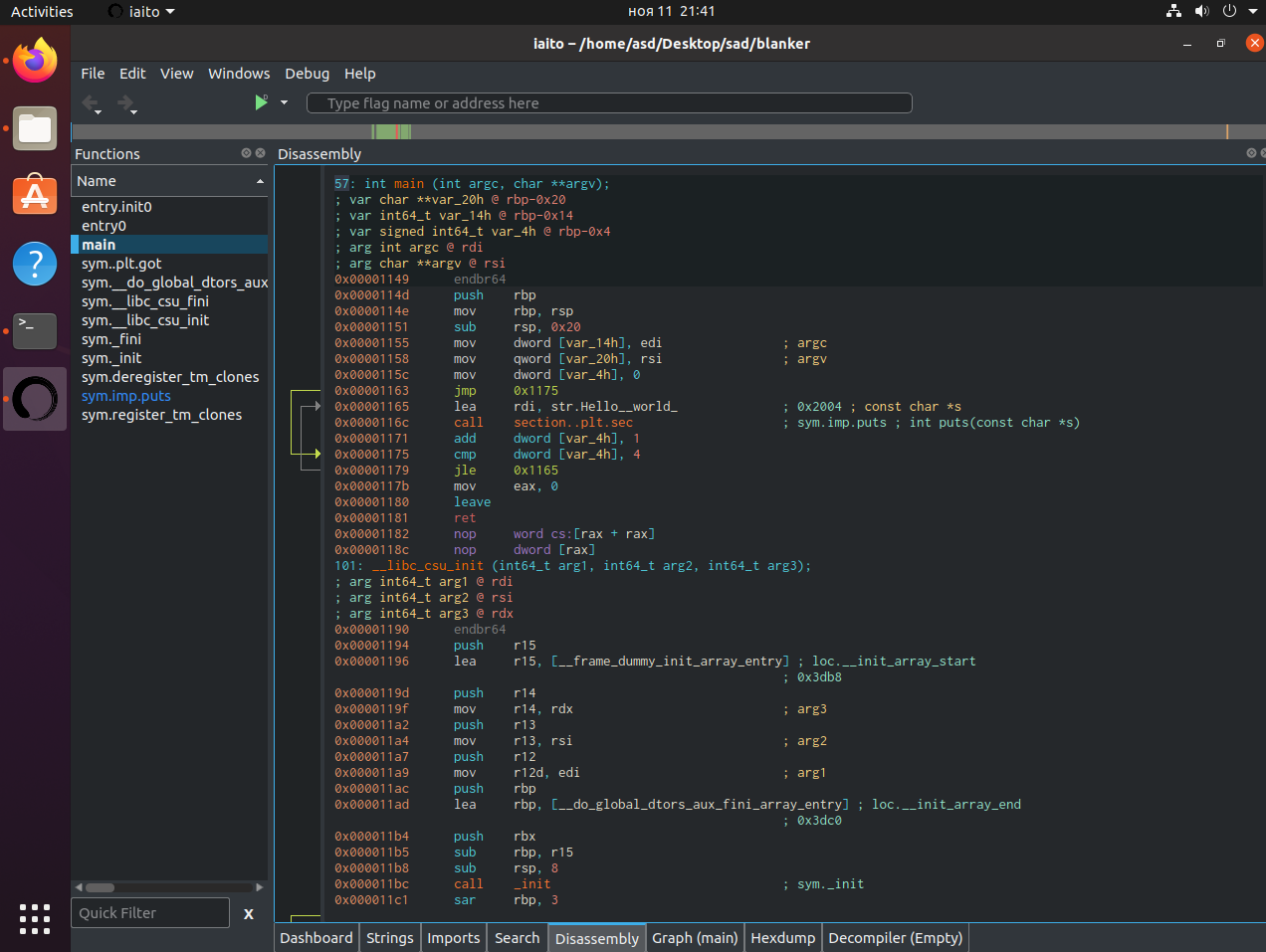


Рис.15

Открываем вкладку декомпилятора. Видим, что код практически читаем, кроме переменных

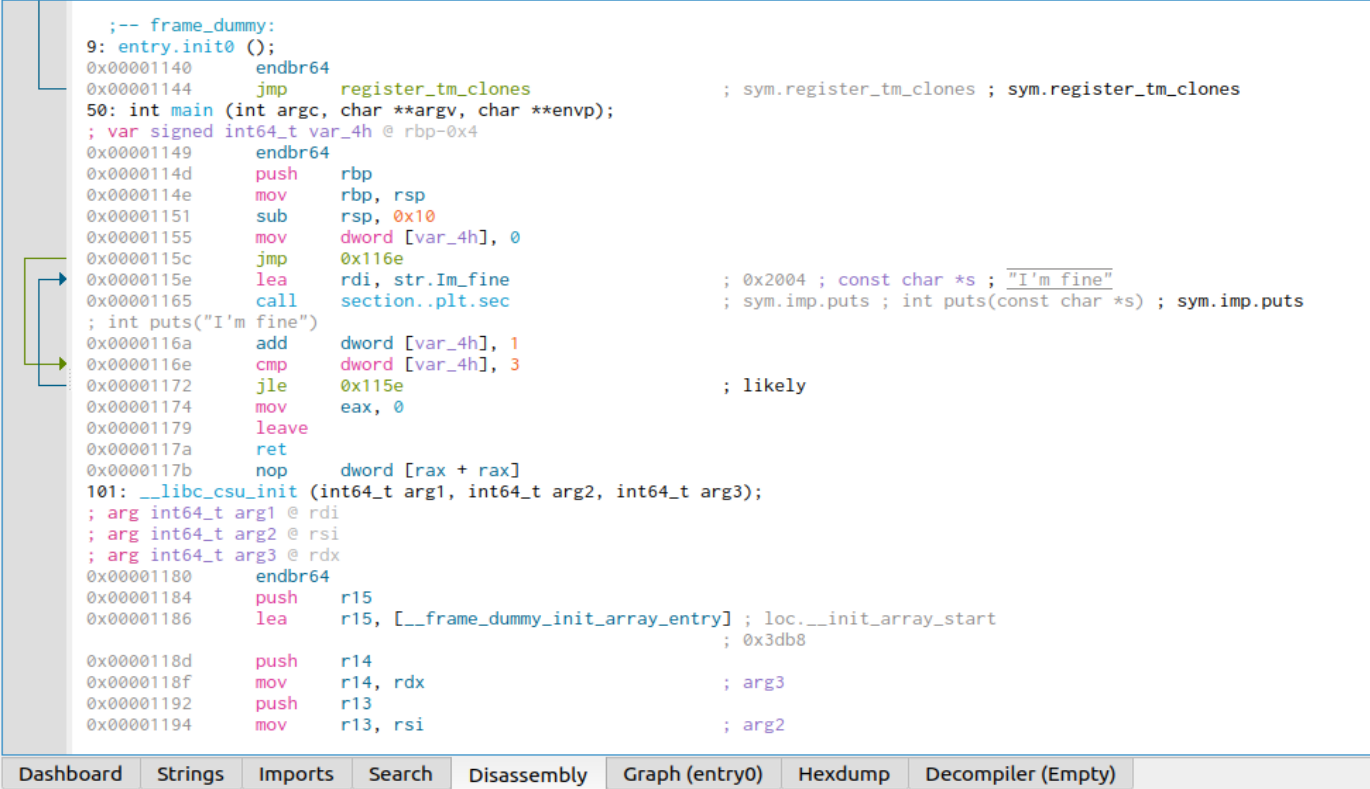


Рис.16

Меняем режим работы программы (File > Set mode > Cache mode)

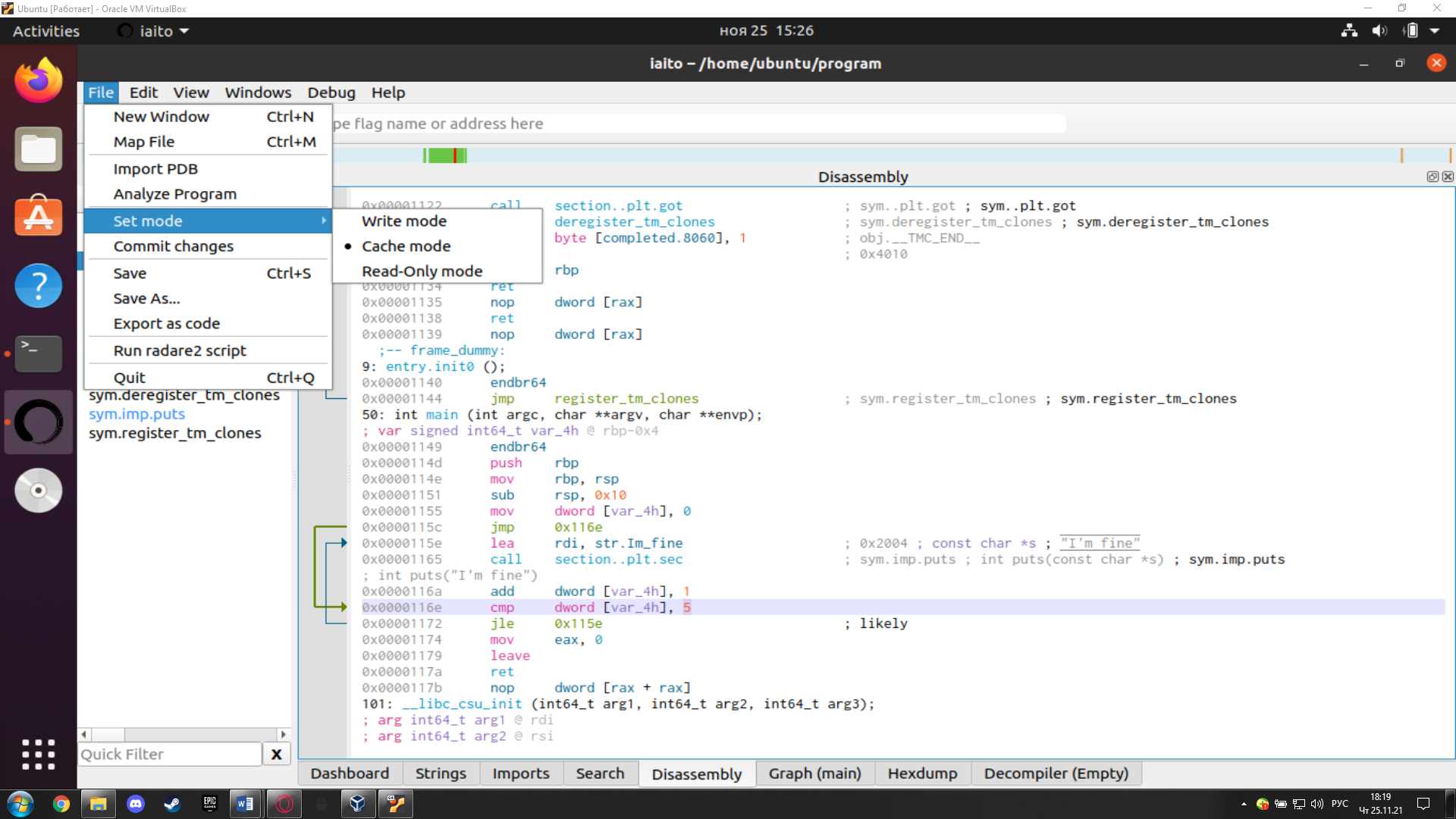


Рис.17

Так как фраза «I’m fine» выводится 5 раз, а нам необходимо увеличить вывод до 10 раз, то смотря на код, можно увидеть, что используется цикл «while» по значение переменной [var4\_h] равной 4 (т.к While <=4, а цикл начинается с 0). Выделяем цифру 4 и нажимаем «Edit» меняя значение переменное с 4 до 10.

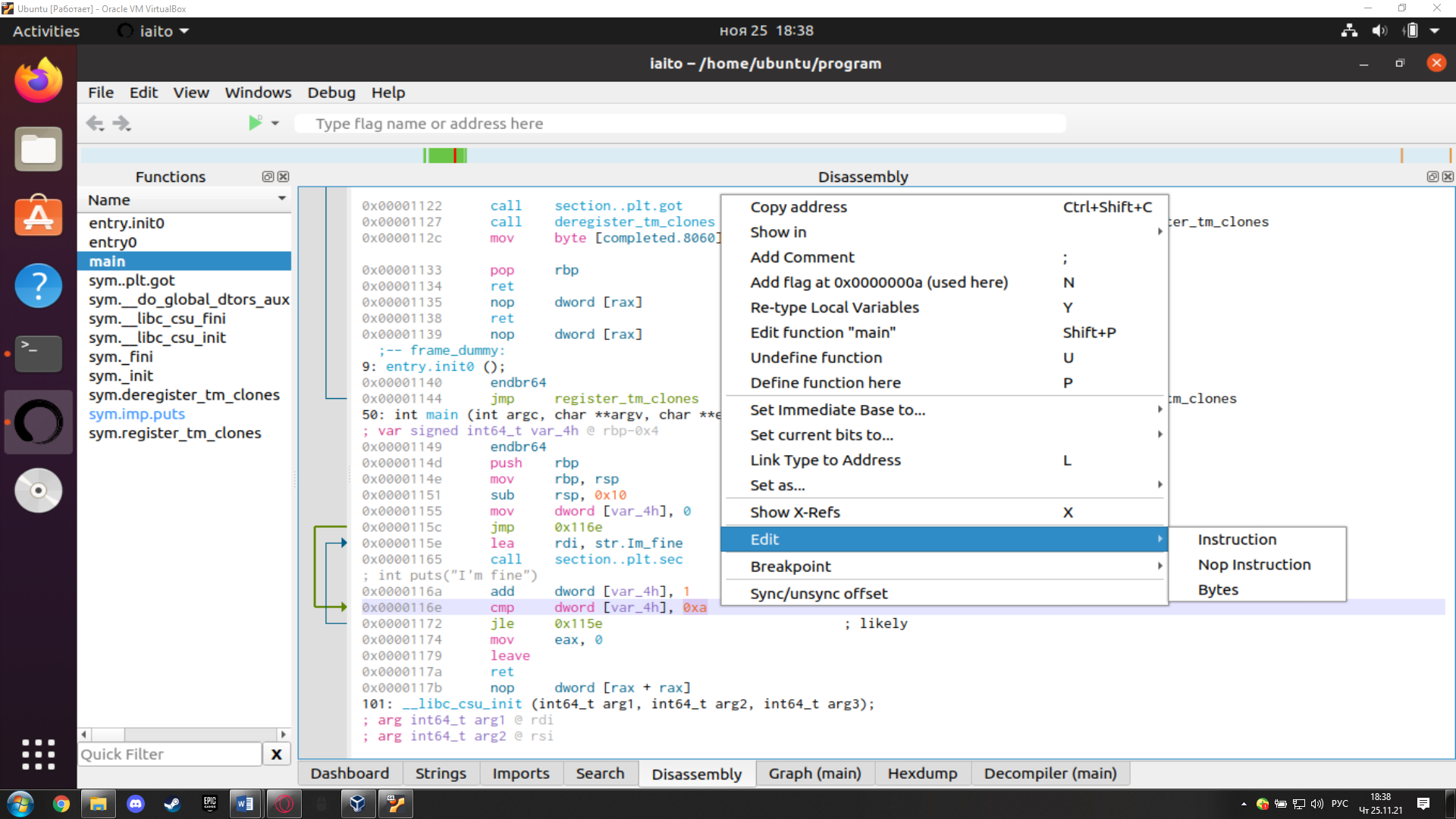


Рис.18

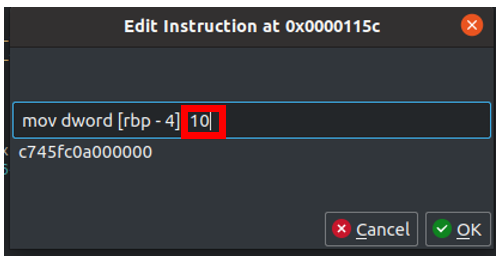


Рис.19

Сохраняем наши изменения, нажав на кнопку «Commit changes»

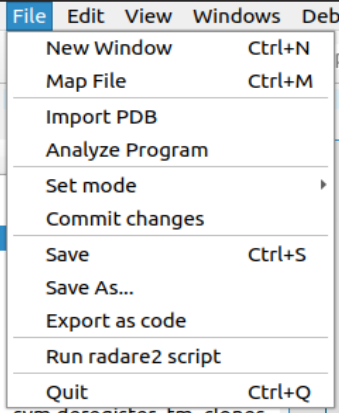


Рис.20

Запускаем наш файл для проверки

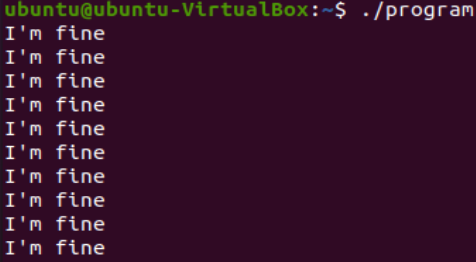


Рис.21