Отчёт по лабораторной работе №10

Тарутина Кристина Олеговна

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

Создаю каталог для выполнения лабораторной работы № 10, перехожу в него и создаю файл lab10-1.asm(рис. 1)



Рис. 1: Создание каталога и файла

Ввожу в файл текст их листинга 10.1(рис. 2)

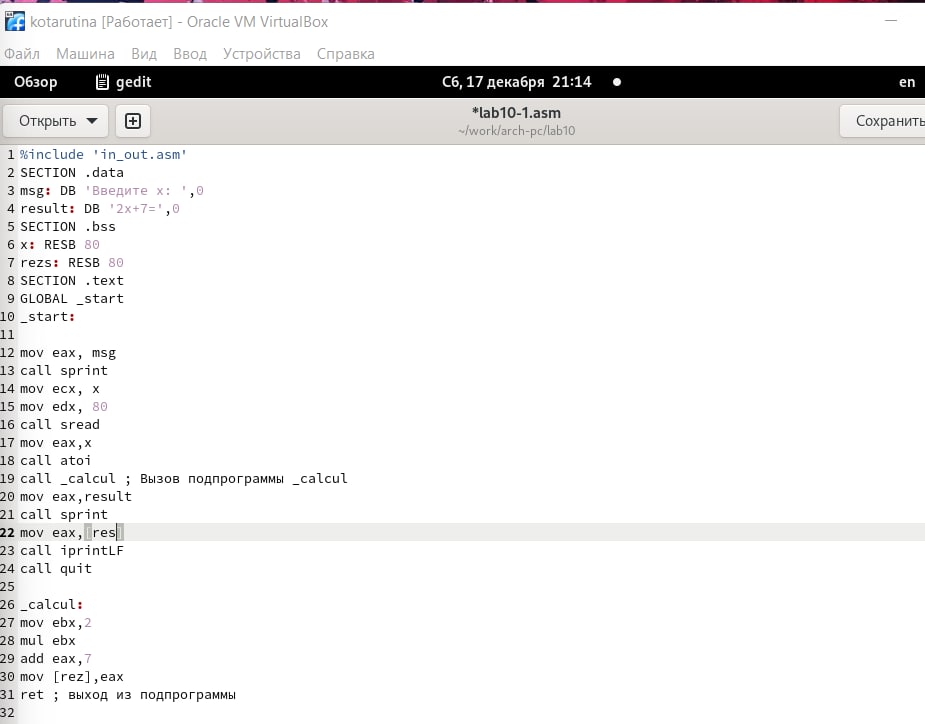


Рис. 2: Текст программы

Создаю исполняемый файл и проверяю его работу. Программа не работает из-за ошибок в обозначении(рис. 3)

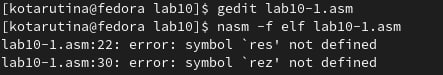


Рис. 3: Работа программы

Нахожу ошибки. Заменяю rezs на rez и res на rez(рис. 4)

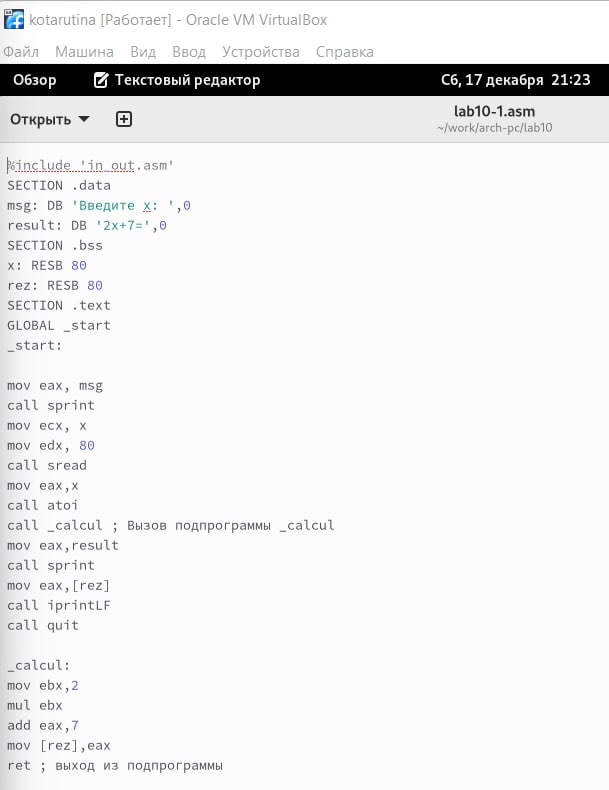


Рис. 4: Исправленный текст программы

Снова создаю исполняемый файл и запускаю. Программа работает корректно(рис. 5)

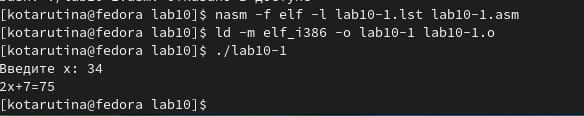


Рис. 5: Работа исправленной программы

Изменяю текст программы, добавив подпрограмму \_subcalcul в подпрограмму \_calcul(рис. 6) создаю исполняемый файл и проверяю его работу (рис. 7)

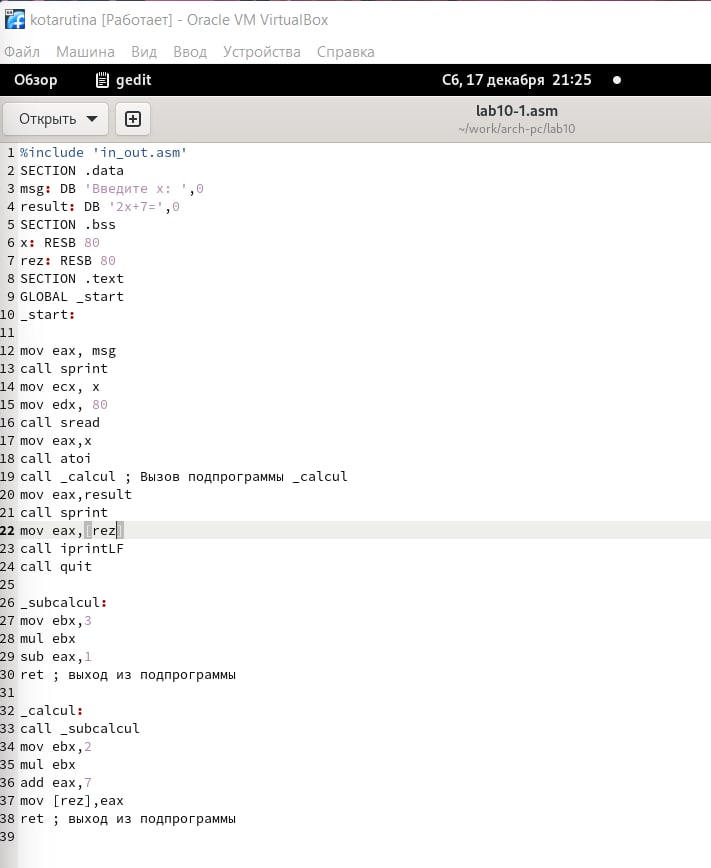


Рис. 6: Текст изменённой программы

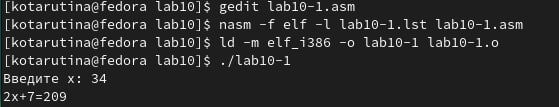


Рис. 7: Работа программы

Создаю файл lab10-2.asm с текстом программы из Листинга 10.2, получаю исполняемый файл и загружаю его в отладчик gdb8 - 9)

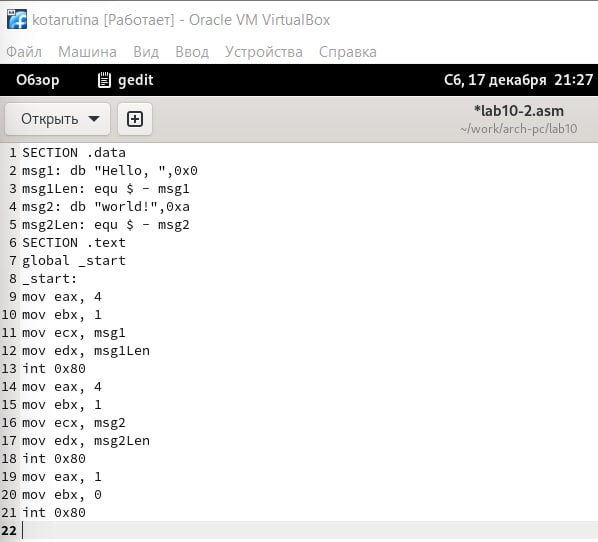


Рис. 8: Текст программы

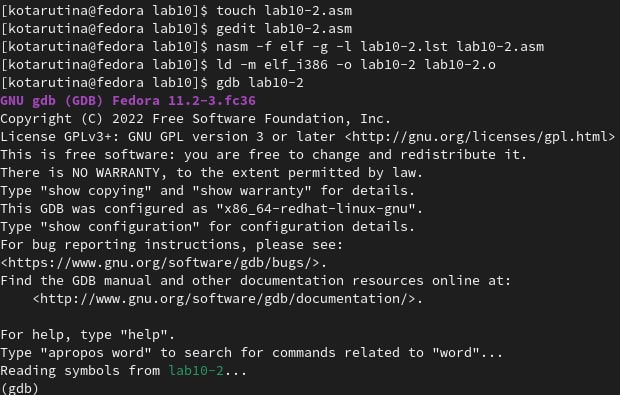


Рис. 9: Создание исполняемого файла и загрузка в отладчик

Запускаю программу в оболочке GDB с помощью команды run(рис. 10)

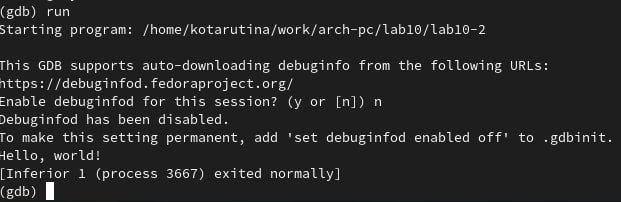


Рис. 10: Действие команды run

Устанавливаю брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запускаю её. (рис. 11)



Рис. 11: Установка брейкпоинта на метку

Смотрю дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. 11)



Рис. 12: Дисассимилированный код программы

Переключаюсь на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel(рис. 13)

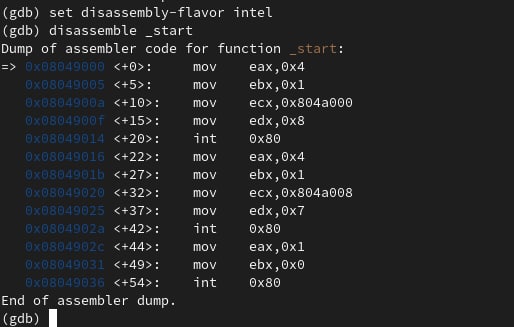


Рис. 13: команды с Intel’овским синтаксисом

Различия в синтаксисах в том, что в первом отображаются спецсимволы в четвёртой колонке(где отображаются переменные), а во втором нет

Включаю режим псевдографики для более удобного анализа программы (рис. 14)

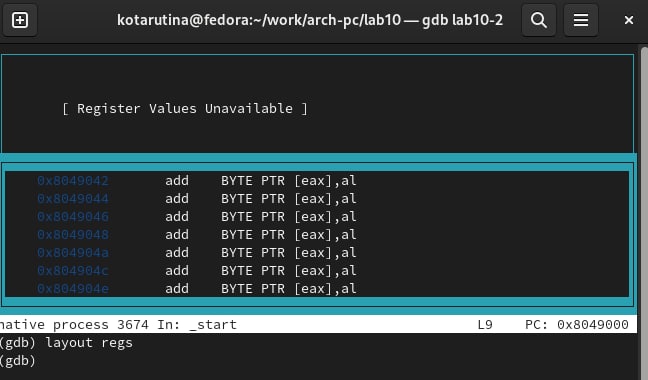


Рис. 14: режим псевдографики

На предыдущих шагах была установлена точка останова по имени метки (\_start). Проверяю это с помощью команды info breakpoints(рис. 15)



Рис. 15: Действие команды info breakpoints

Определяю адрес предпоследней инструкции (mov ebx,0x0) и установливаю точку останова. Смотрю информацию о всех установленных точках останова(рис. 16)

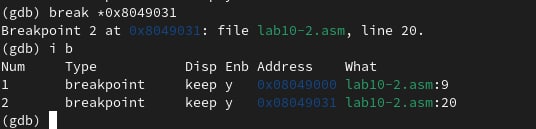


Рис. 16: Установка точки

Выполняю 5 инструкций с помощью команды stepi (или si) Изменяются регистры eax, ebx, ecx, edx Смотрю содержимое регистров с помощью команды info registers(рис. 17 - 18)

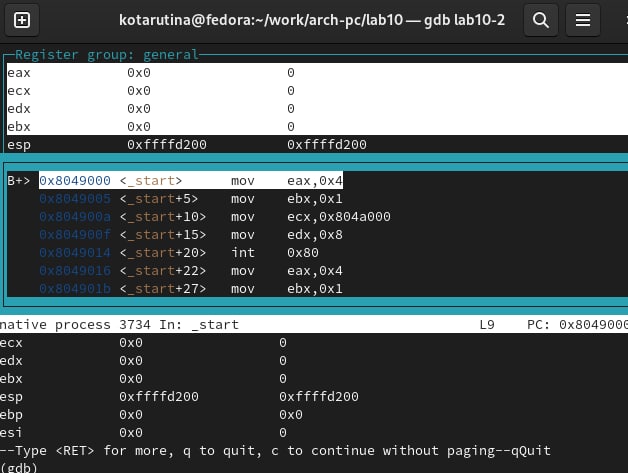


Рис. 17: Содержимое регистров

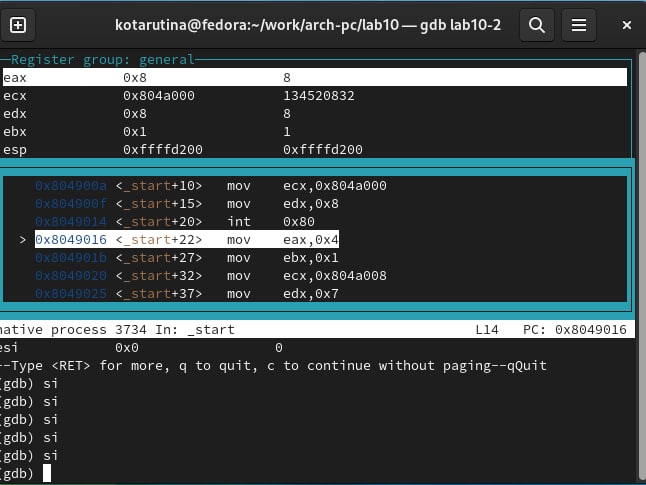


Рис. 18: Команда si

Смотрю значение переменной msg1 по имени и значение переменной msg2 по адресу(рис. 19)

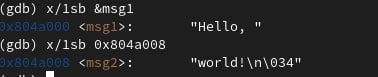


Рис. 19: значение переменных

Изменяю первый символ переменной msg1 и msg2(рис. 20)

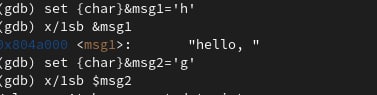


Рис. 20: изменение переменных

С помощью команды set изменяю значение регистра ebx(рис. 21)

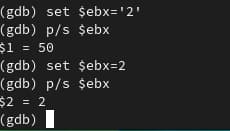


Рис. 21: значение регистра ebx

КОпирую файл lab9-2.asm, созданный при выполнении лабораторной работы №9, создаю тсполняемый файл(рис. 22)

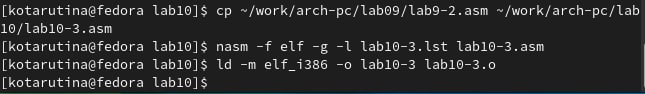


Рис. 22: копирование файла и создание исполняемого

Загружаю исполняемый файл в отладчик, указав аргументы(рис. 23)

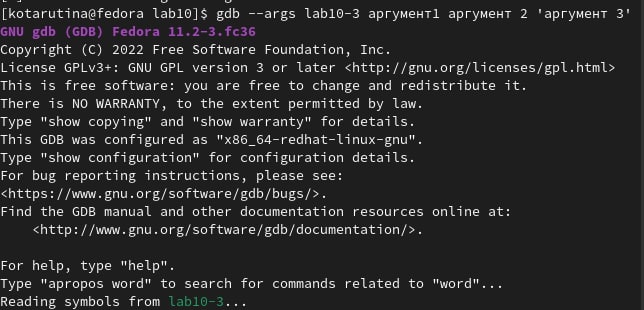


Рис. 23: загрузка исполняемого файла в отладчик

УСтанавливаю точку останова перед первой инструкцией в программе и запускаю ее.(рис. 24)

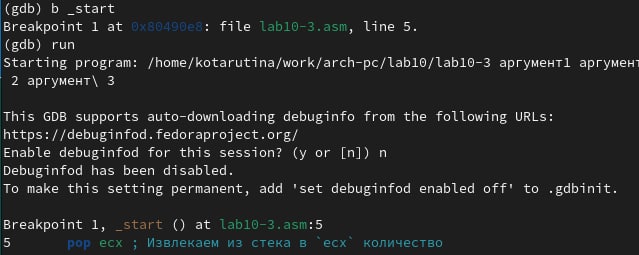


Рис. 24: УСтановка точки и запуск программы

Просматриваю позиции стеков(рис. 25 - 26)

Рис. 25: 25

Рис. 25: 25

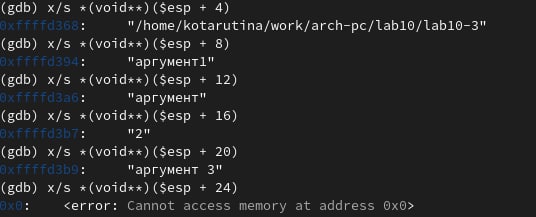


Рис. 26: 26

Размер переменной будет четыре байта. А шаг равен размеру переменной

# 3 Выполнение самостоятельной работы

Преобразую программу из лабораторной работы номер 9(мой вариант 9)(рис. 27 - 28)

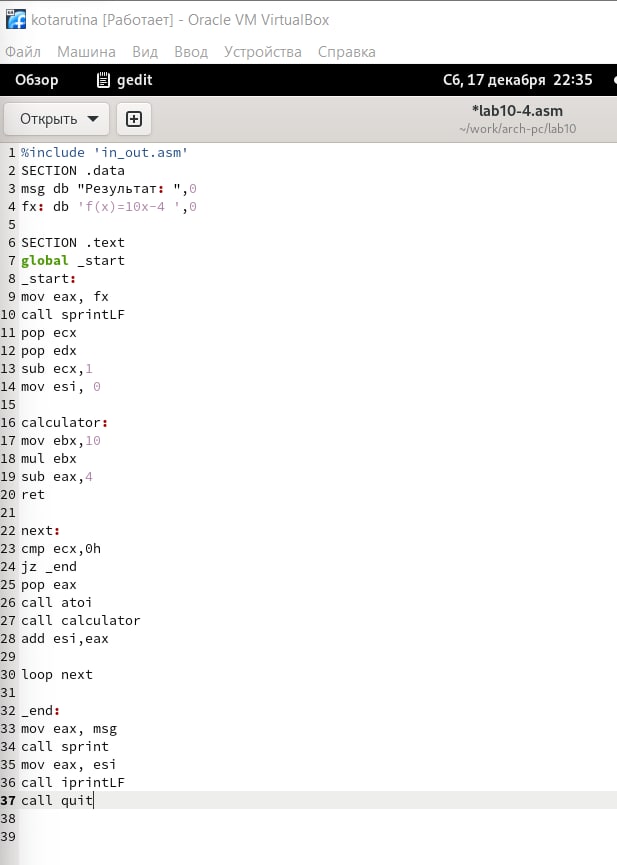


Рис. 27: текст программы

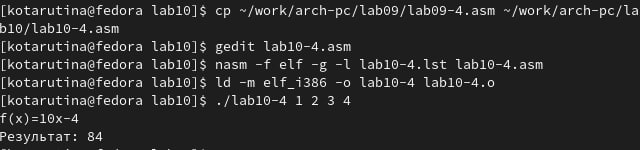


Рис. 28: Работа программы

Создаю файл и записываю в него текст программы. Проверяю работ. Программа выводит неверный ответ. Загружаю файл в отладчик(рис. 29)



Рис. 29: Работа программы

Устанавливаю точку и запускаю программу(рис. 30)



Рис. 30: УСтановка точки и запуск

Делаю первые три шага, не замечаю ошибки на данном этапе(рис. 31)

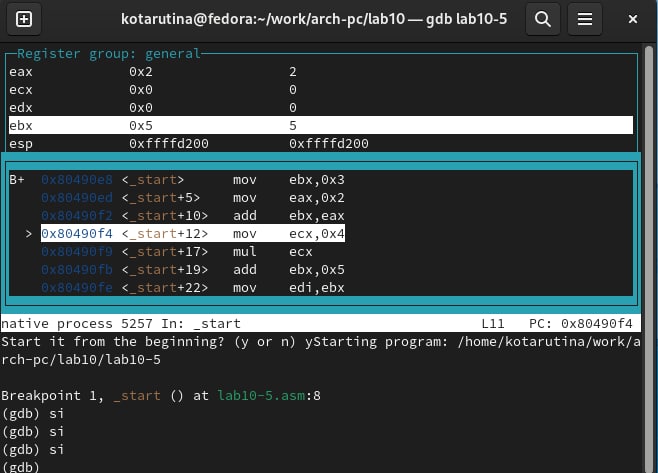


Рис. 31: три шага в отладчике

Делаю ещё два шага. Программа умножает между собой неверные переменные( должно быть (3+2)*4, но происходит 2*4)(рис. 32)

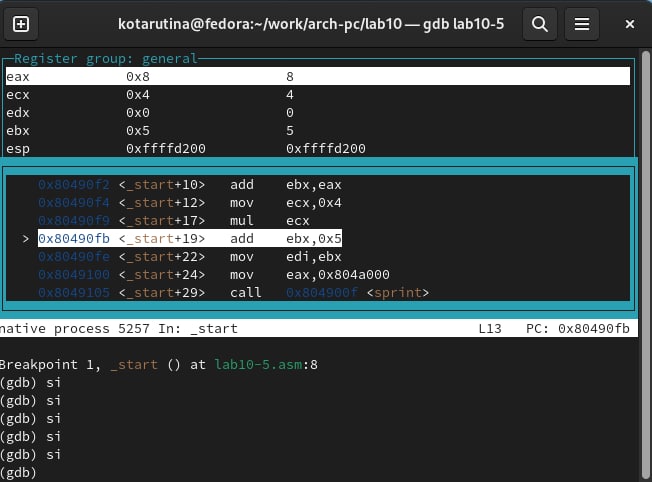


Рис. 32: пять шагов в отладчике

Делаю ещё один шаг. ПО итогу программа умножает неверные переменные и прибавляет пять к переменной, которая не была умножена на 4(рис. 33)

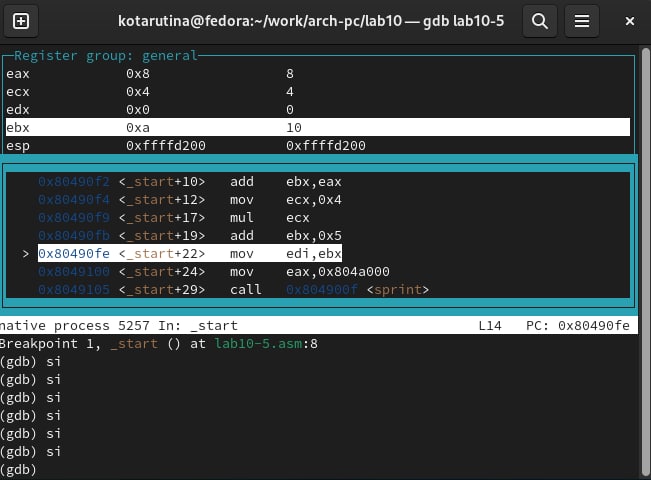


Рис. 33: шесть шагов в отладчике

Исправляю ошибку в программе(рис. 34)

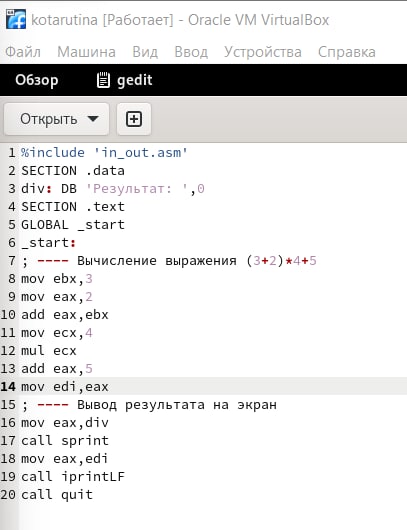


Рис. 34: ИСправленный текст программы

Создаю исполняемый файл и проверяю работу программы(рис. 35)

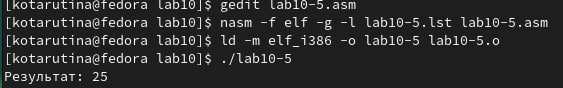


Рис. 35: РАбота программы

# 4 Выводы

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