**Отчёта по лабораторной работе 10**

### Понятие подпрограммы. Отладчик GDB.

## Герра Гарсия Паола Валентина НКАбд-05-22

**Содержание**

1. [Цель работы](#_bookmark0) 5
2. [Задание](#_bookmark1) 6
3. [Теоретическое введение](#_bookmark2) 7
4. [Выполнение лабораторной работы](#_bookmark3) 8
5. [Выводы](#_bookmark25) 30

[Список литературы](#_bookmark26) 31

**Список иллюстраций**

[4.1 Файл lab10-1.asm](#_bookmark4) . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

* 1. [Работа программы lab10-1.asm](#_bookmark5) 10
  2. [Файл lab10-1.asm](#_bookmark6) 11
  3. [Работа программы lab10-1.asm](#_bookmark7) 12
  4. [Файл lab10-2.asm](#_bookmark8) 13
  5. [Работа программы lab10-2.asm в отладчике](#_bookmark9) 14
  6. [дисассимилированный код](#_bookmark10) 15
  7. [дисассимилированный код в режиме интел](#_bookmark11) 16
  8. [точка остановки](#_bookmark12) 17
  9. [изменение регистров](#_bookmark13) 18
  10. [изменение регистров](#_bookmark14) 19
  11. [изменение значения переменной](#_bookmark15) 20
  12. [вывод значения регистра](#_bookmark16) 21
  13. [вывод значения регистра](#_bookmark17) 22
  14. [вывод значения регистра](#_bookmark18) 23
  15. [Файл lab10-4.asm](#_bookmark19) 24
  16. [Работа программы lab10-4.asm](#_bookmark20) 25
  17. [код с ошибкой](#_bookmark21) 26
  18. [отладка](#_bookmark22) 27
  19. [код исправлен](#_bookmark23) 28
  20. [проверка работы](#_bookmark24) 29

**Список таблиц**

# Цель работы

Целью работы является приобретение навыков написания программ с исполь- зованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

# Задание

* 1. Изучите примеры реализации подпрограмм
  2. Изучите работу с отладчиком GDB
  3. Выполните самостоятеьное задание
  4. Загрузите файлы на GitHub.

# Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа:

* обнаружение ошибки;
* поиск её местонахождения;
* определение причины ошибки;
* исправление ошибки.

# Выполнение лабораторной работы

* 1. Создайте каталог для выполнения лабораторной работы № 10, перейдите в него и создайте файл lab10-1.asm:
  2. 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. Внимательно изучите текст программы (Листинг 10.1). (рис. [4.1,](#_bookmark4) [4.2)](#_bookmark5)

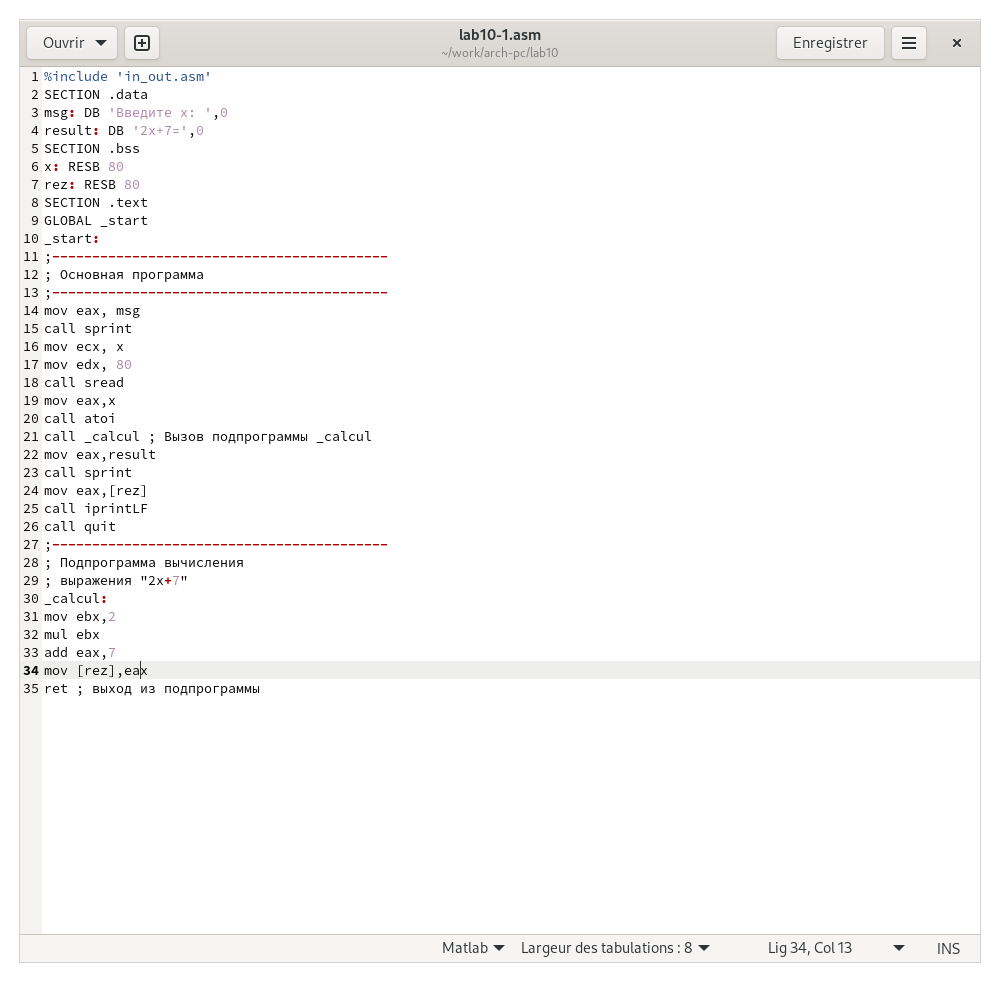


Рис. 4.1: Файл lab10-1.asm

Рис. 4.2: Работа программы lab10-1.asm

* 1. Измените текст программы, добавив подпрограмму subcalcul в подпрограм- му calcul, для вычисления выражения f(g(x)), где x вводится с клавиатуры, f(x) = 2x + 7, g(x) = 3x − 1(рис. [4.3,](#_bookmark6) [4.4)](#_bookmark7)

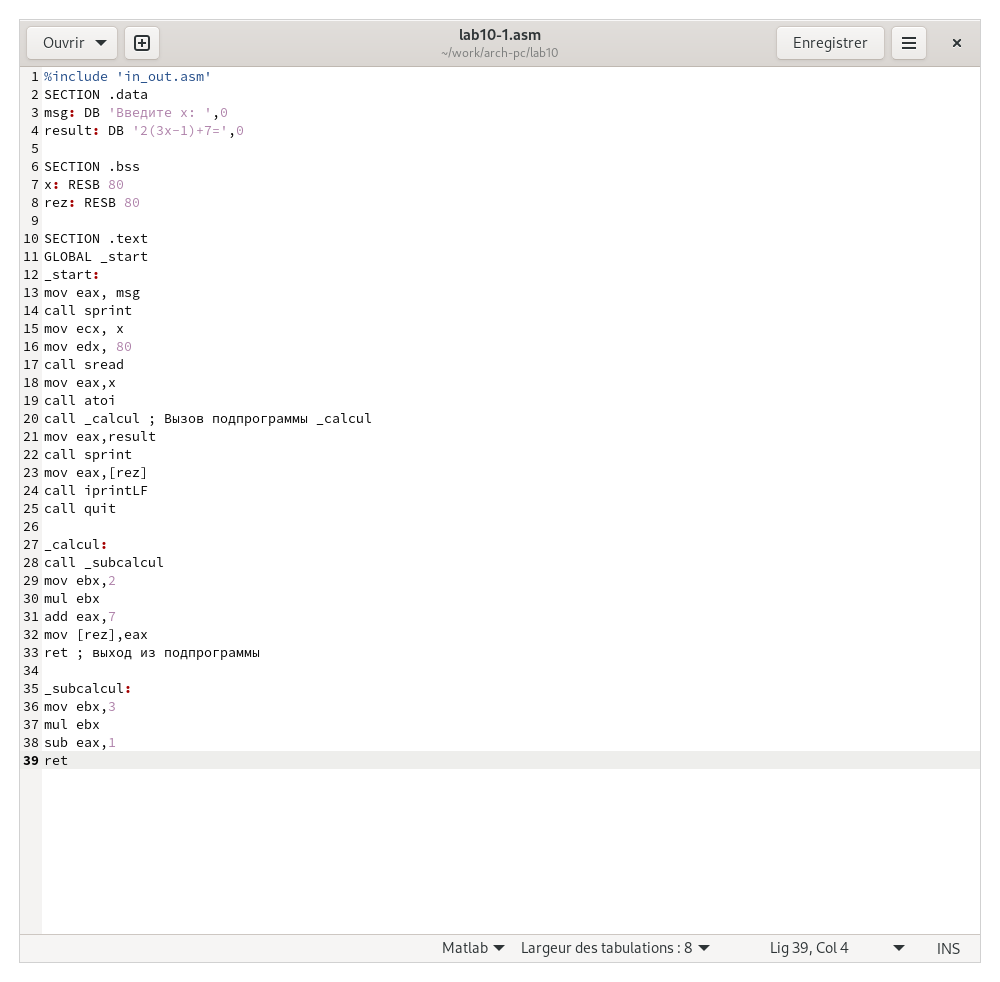


Рис. 4.3: Файл lab10-1.asm

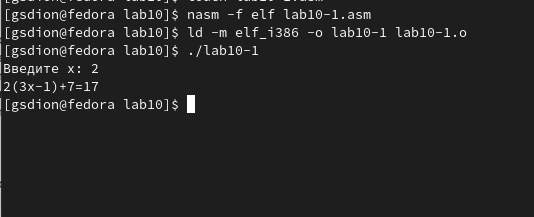
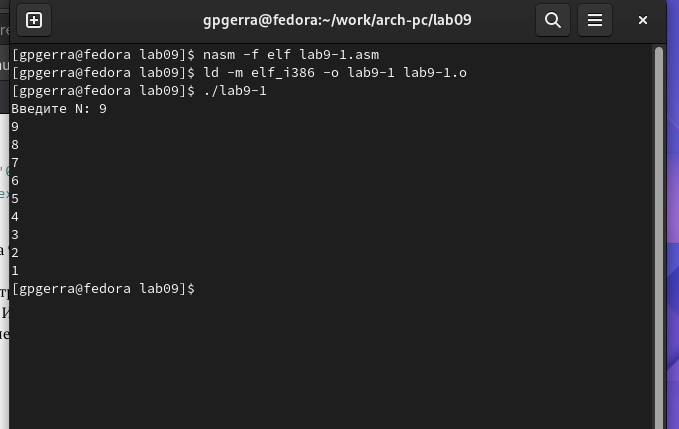
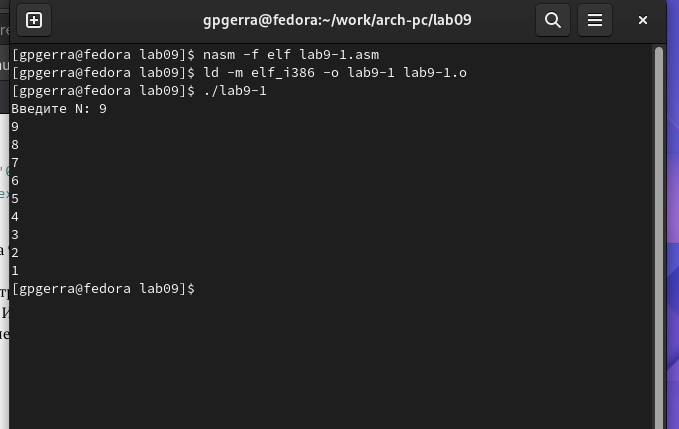
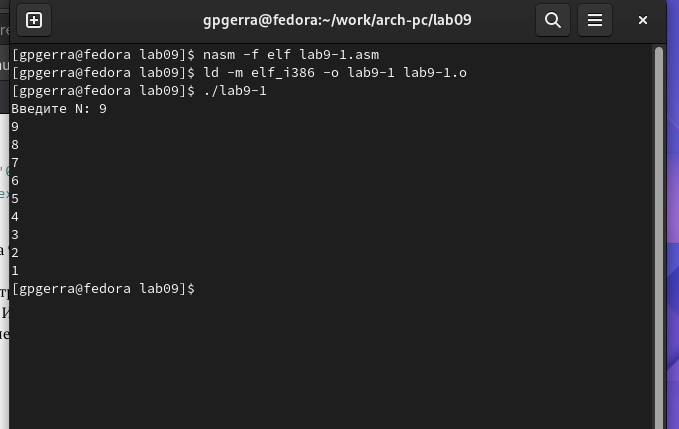
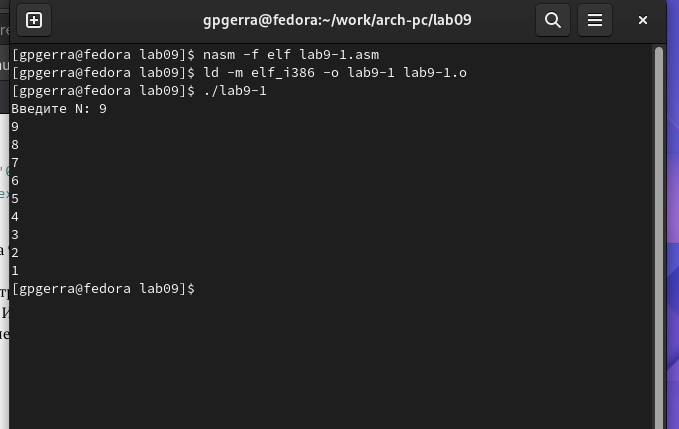


Рис. 4.4: Работа программы lab10-1.asm

* 1. Создайте файл lab10-2.asm с текстом программы из Листинга 10.2. (Про- грамма печати сообщения Hello world!): (рис. [4.5)](#_bookmark8)



Рис. 4.5: Файл lab10-2.asm

Получите исполняемый файл. Для работы с GDB в исполняемый файл необ- ходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’. Загрузите исполняемый файл в отладчик gdb: Проверьте работу программы, запустив ее в оболочке GDB с помощью ко- манды run (сокращённо r):(рис. [4.6)](#_bookmark9)

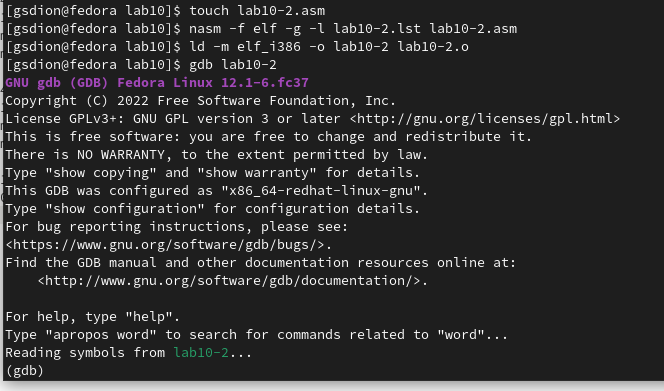


Рис. 4.6: Работа программы lab10-2.asm в отладчике

Для более подробного анализа программы установите брейкпоинт на мет- ку start, с которой начинается выполнение любой ассемблерной программы, и запустите её. Посмотрите дисассимилированный код программы (рис. [4.7,](#_bookmark10) [4.8)](#_bookmark11)

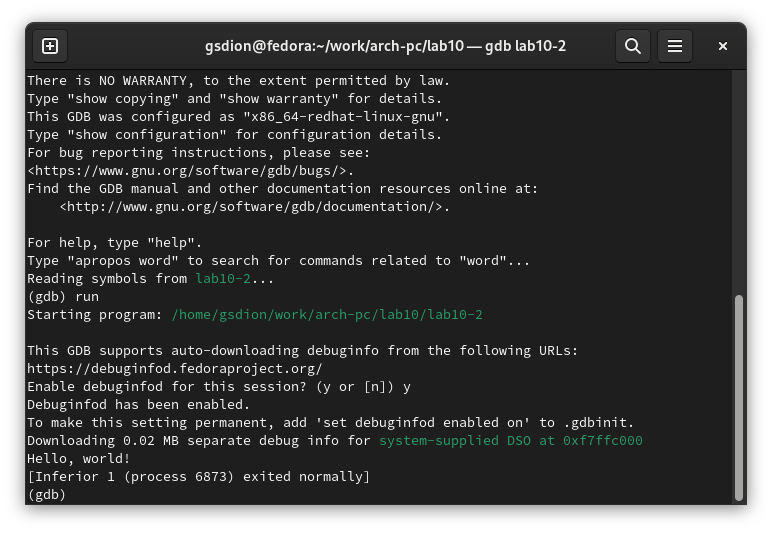
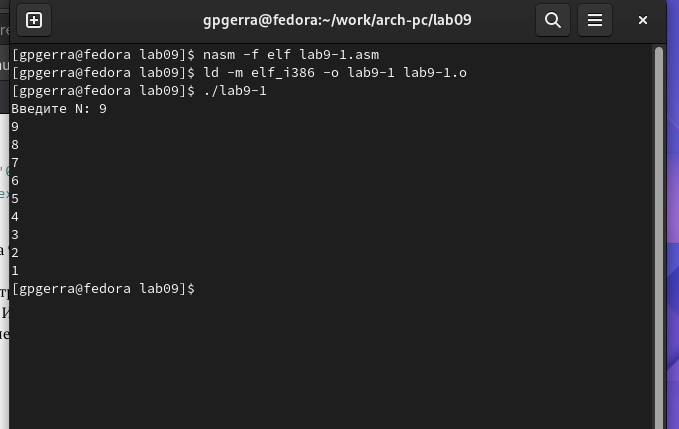


Рис. 4.7: дисассимилированный код

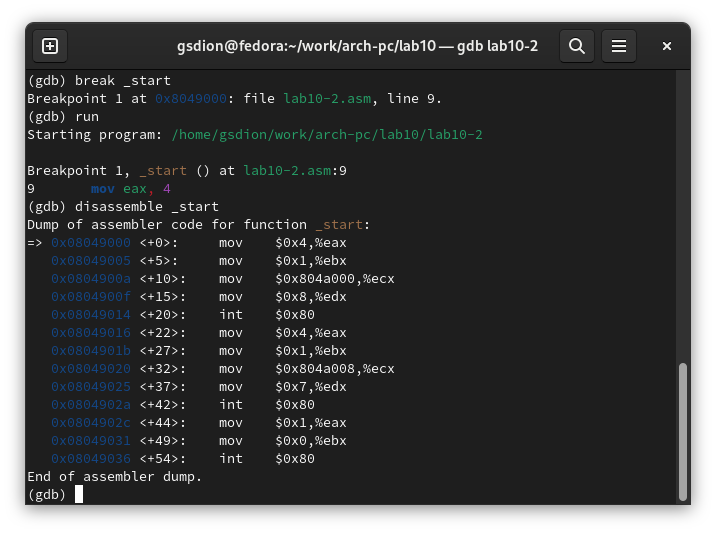
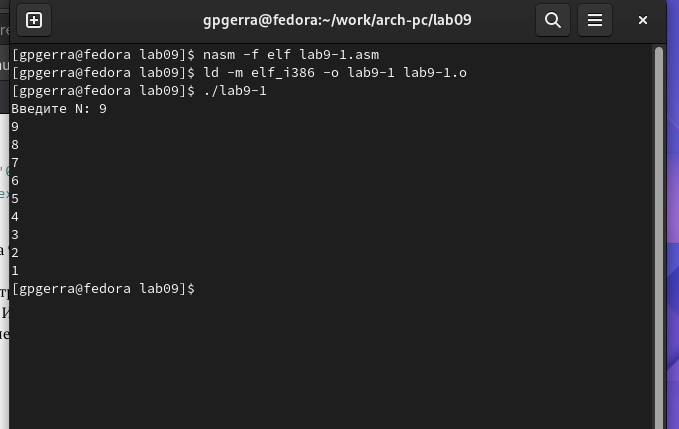


Рис. 4.8: дисассимилированный код в режиме интел

На предыдущих шагах была установлена точка останова по имени метки (\_start). Проверьте это с помощью команды info breakpoints (кратко i b) Установим еще одну точку останова по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции. Определите адрес предпоследней инструкции (mov ebx,0x0) и установите точку.(рис. [4.9)](#_bookmark12)

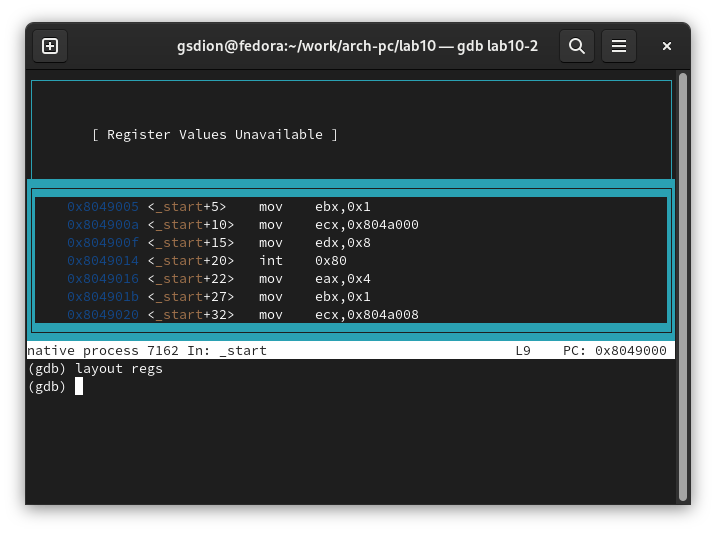
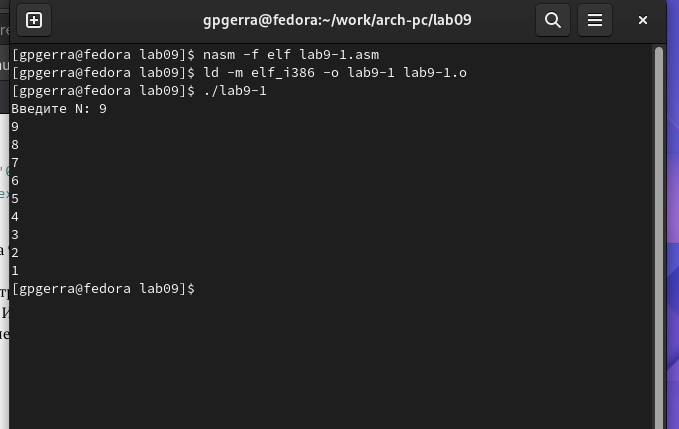


Рис. 4.9: точка остановки

Отладчик может показывать содержимое ячеек памяти и регистров, а при необходимости позволяет вручную изменять значения регистров и переменных. Выполните 5 инструкций с помощью команды stepi (или si) и проследите за изменением значений регистров. (рис. [4.11](#_bookmark14) [4.12)](#_bookmark15)

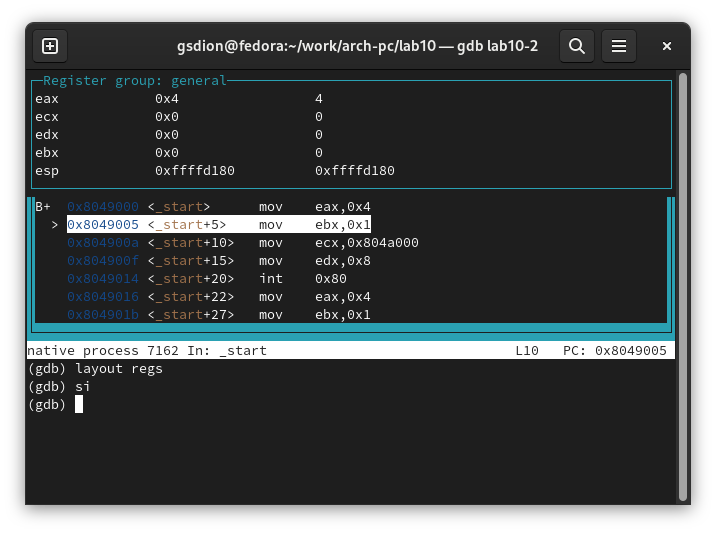
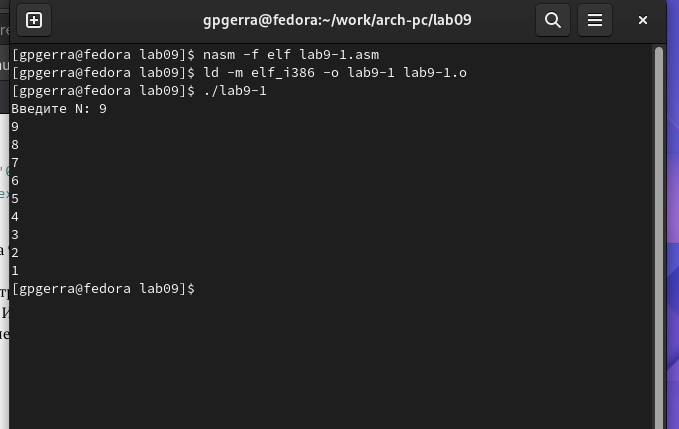


Рис. 4.10: изменение регистров

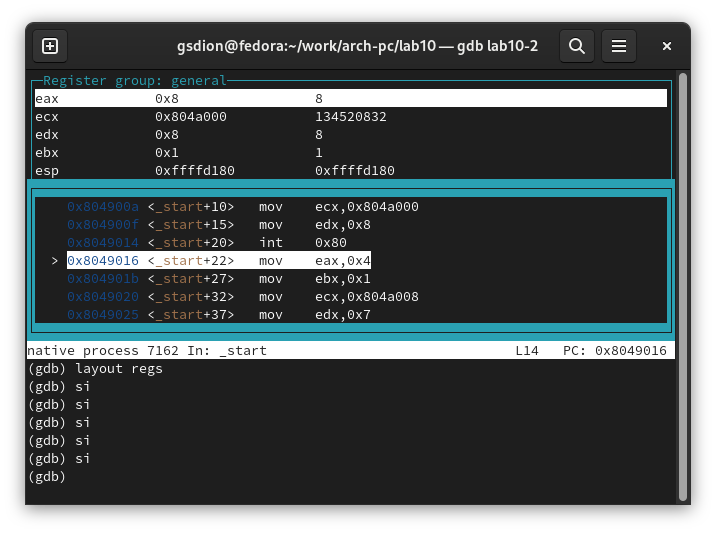
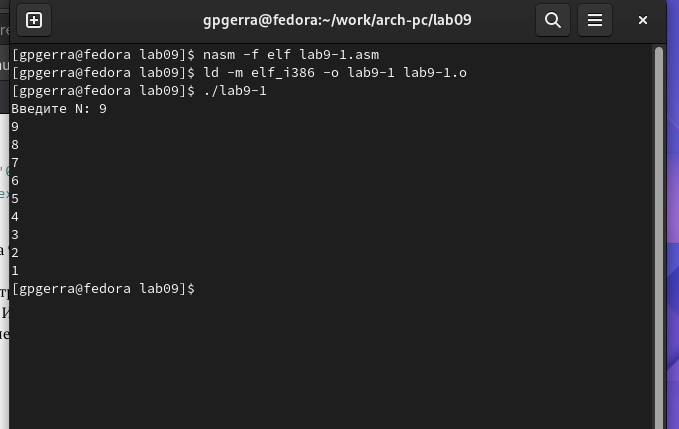


Рис. 4.11: изменение регистров

Посмотрите значение переменной msg1 по имени Посмотрите значение пе- ременной msg2 по адресу Изменить значение для регистра или ячейки памяти можно с помощью команды set, задав ей в качестве аргумента имя регистра или адрес. Измените первый символ переменной msg1 Замените любой символ во второй переменной msg2. (рис. [4.12)](#_bookmark15)

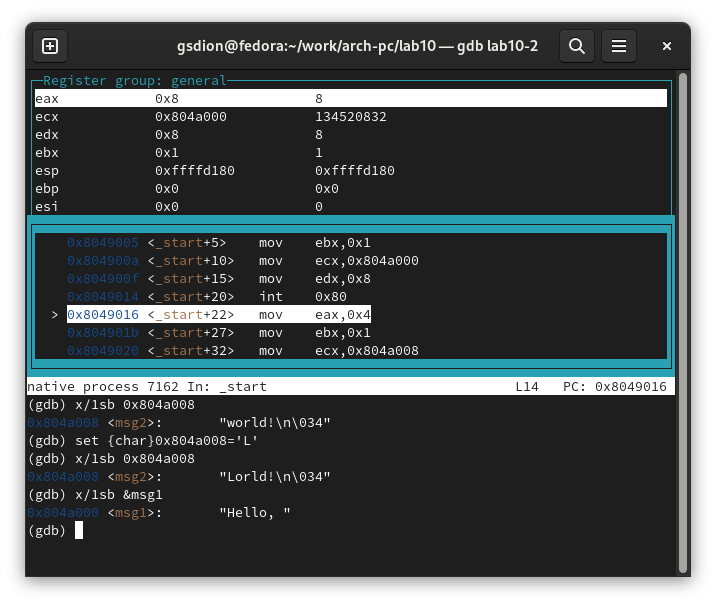
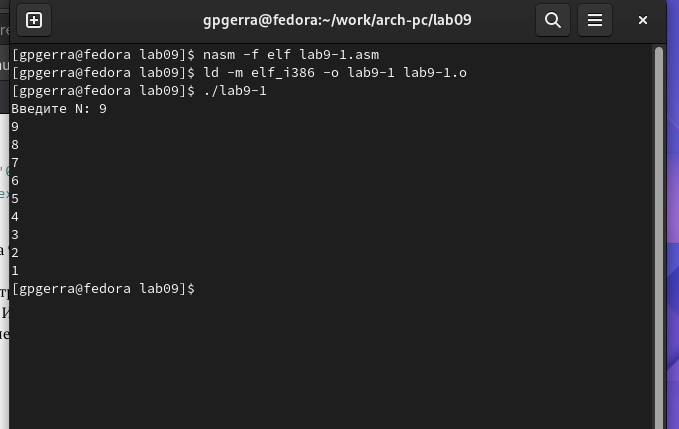


Рис. 4.12: изменение значения переменной

Выведете в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx. С помощью команды set измените значение регистра ebx:(рис. [4.13)](#_bookmark16)

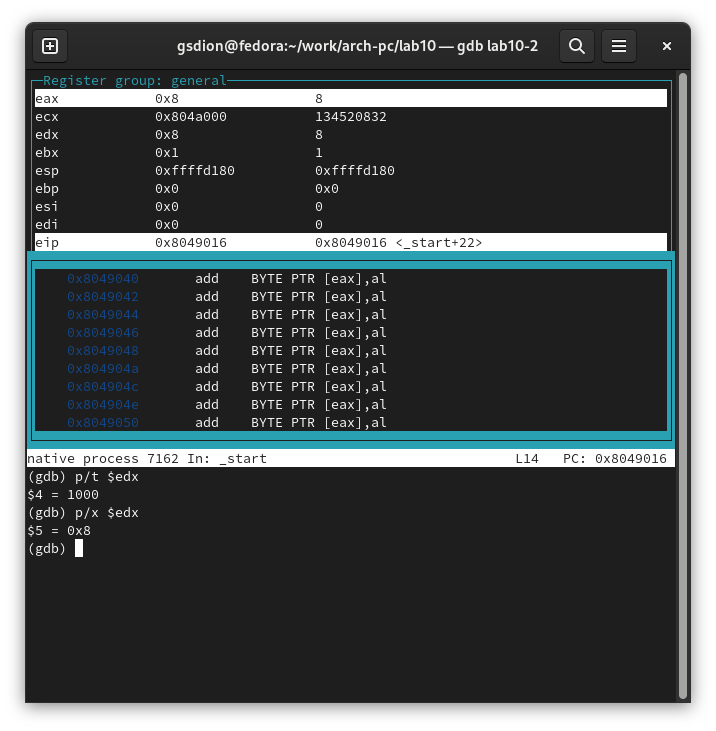
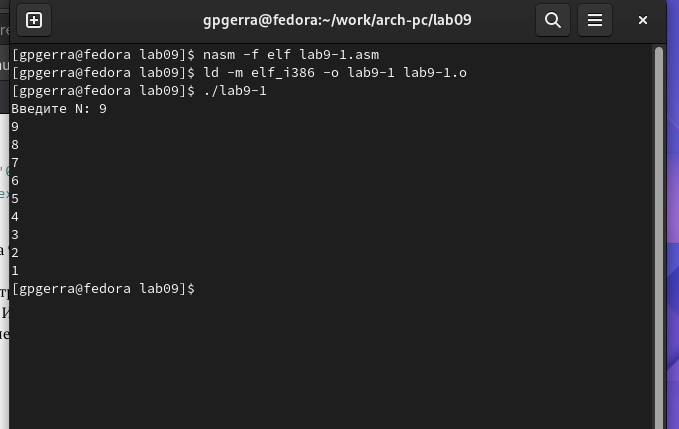


Рис. 4.13: вывод значения регистра

С помощью команды set измените значение регистра ebx:(рис. [4.14)](#_bookmark17)

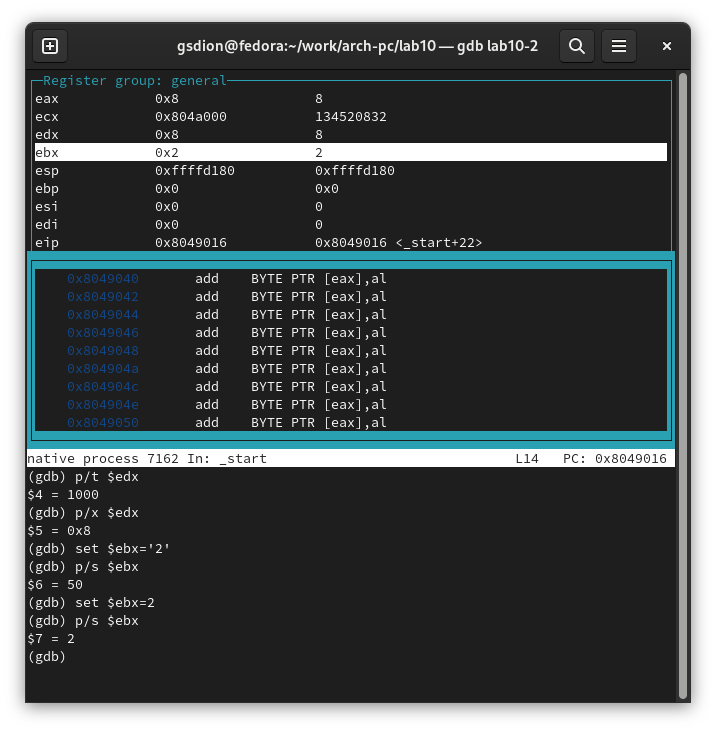
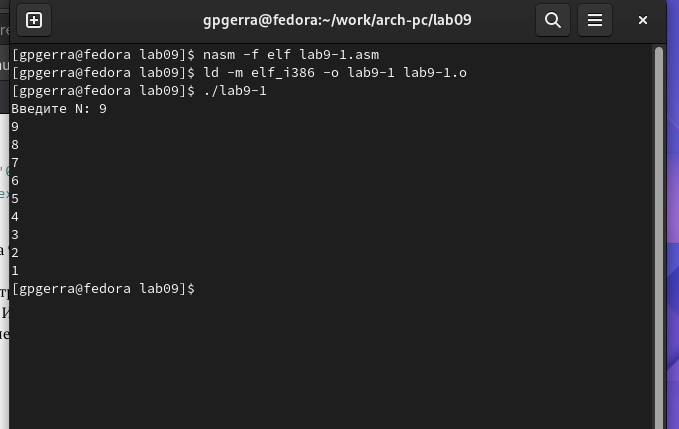


Рис. 4.14: вывод значения регистра

* 1. Скопируйте файл lab9-2.asm, созданный при выполнении лабораторной ра- боты №9, с программой выводящей на экран аргументы командной строки. Создайте исполняемый файл. Для загрузки в gdb программы с аргумента- ми необходимо использовать ключ –args. Загрузите исполняемый файл в отладчик, указав аргументы

Для начала установим точку останова перед первой инструкцией в программе

и запустим ее.

Адрес вершины стека храниться в регистре esp и по этому адресу распола- гается число равное количеству аргументов командной строки (включая имя программы): Как видно, число аргументов равно 5 – это имя программы lab10-3 и непосредственно аргументы: аргумент1, аргумент, 2 и ‘аргумент 3’.

Посмотрите остальные позиции стека – по адесу [esp+4] располагается адрес в памяти где находиться имя программы, по адесу [esp+8] храниться адрес первого аргумента, по аресу [esp+12] – второго и т.д. (рис. [4.15)](#_bookmark18)

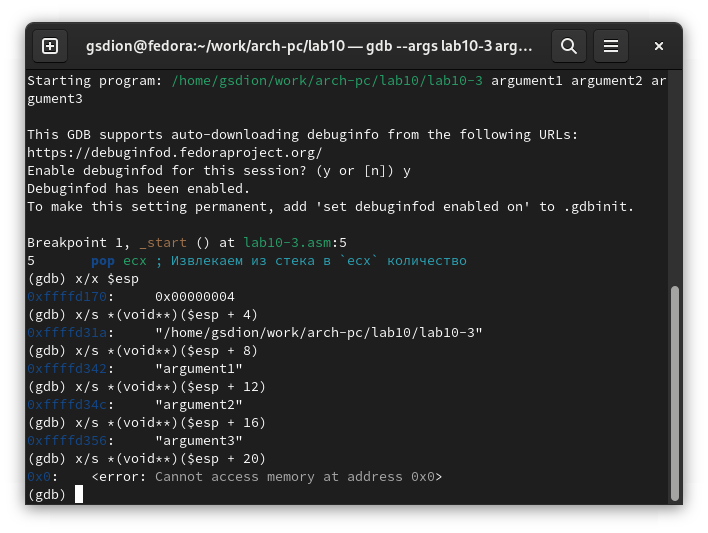
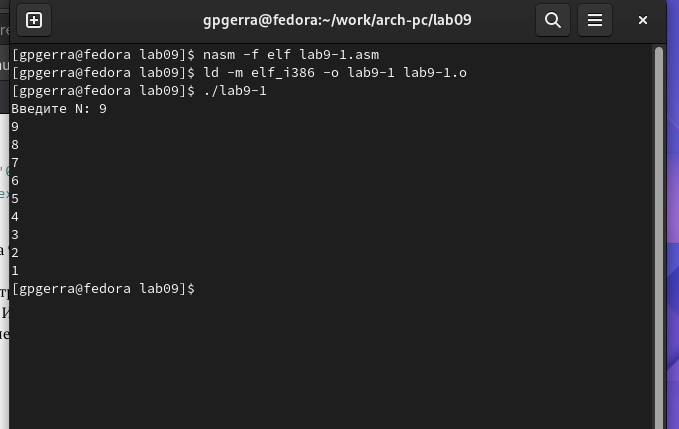


Рис. 4.15: вывод значения регистра

Объясните, почему шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12] - шаг равен размеру переменной - 4 байтам.

* 1. Преобразуйте программу из лабораторной работы №9 (Задание №1 для

самостоятельной работы), реализовав вычисление значения функции f(x) как подпрограмму. (рис. [4.16](#_bookmark19) [4.17)](#_bookmark20)

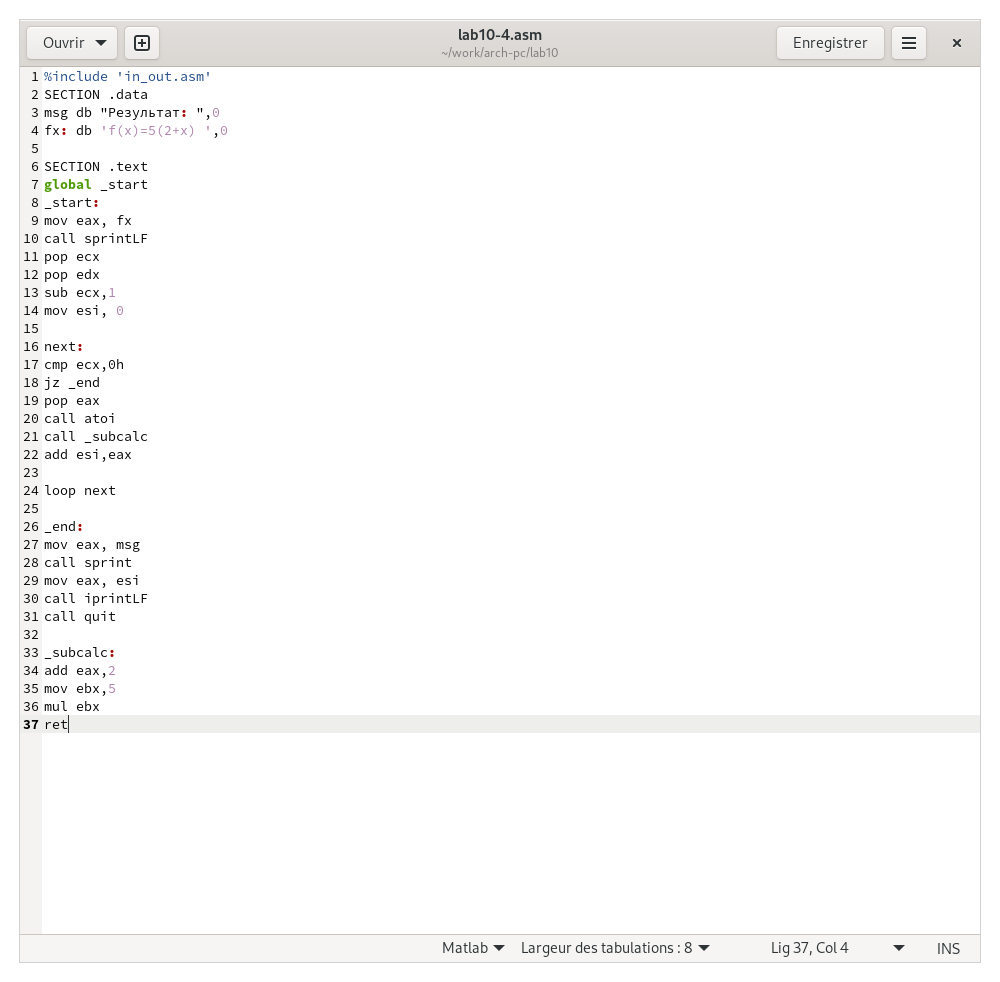


Рис. 4.16: Файл lab10-4.asm

Рис. 4.17: Работа программы lab10-4.asm

* 1. В листинге приведена программа вычисления выражения (3+2) 4+5. При запуске данная программа дает неверный результат. Проверьте это. С по- мощью отладчика GDB, анализируя изменения значений регистров, опре- делите ошибку и исправьте ее.(рис. [4.18](#_bookmark21) [4.19](#_bookmark22) [4.20](#_bookmark23) [4.21)](#_bookmark24)

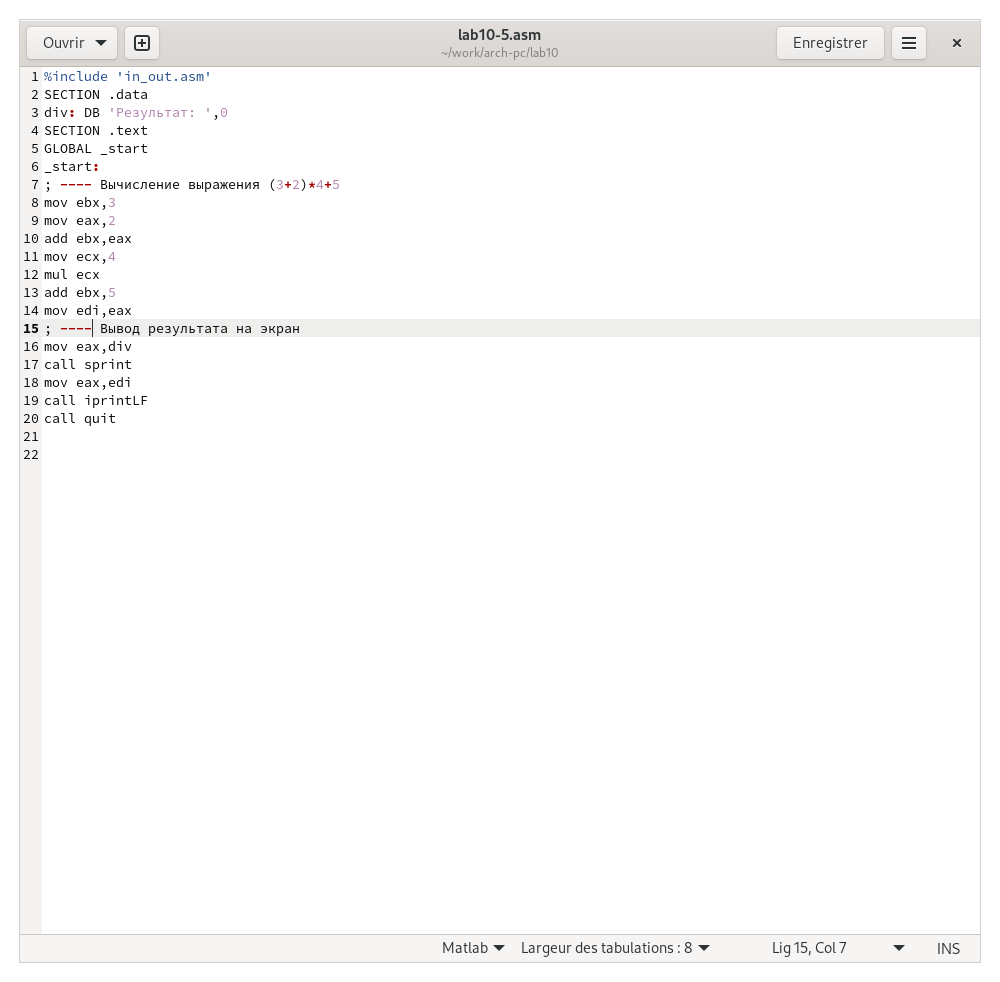


Рис. 4.18: код с ошибкой

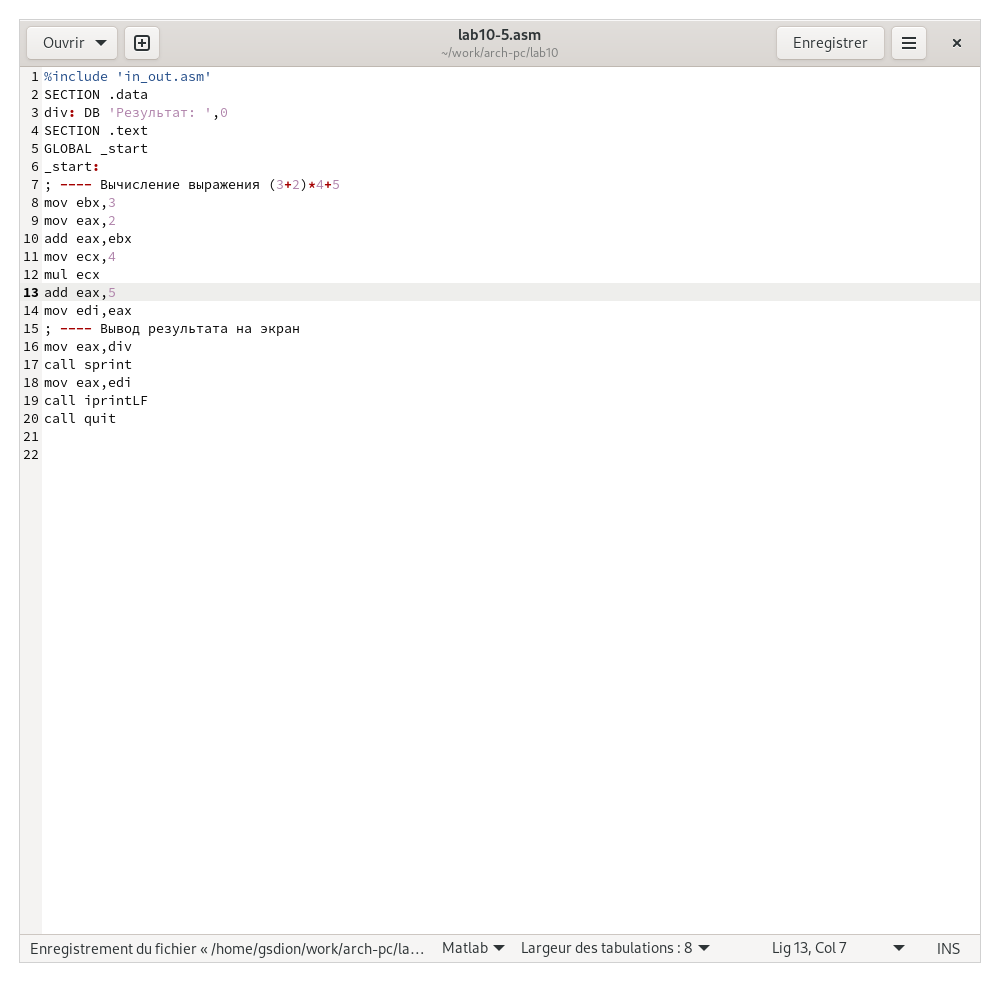


Рис. 4.19: отладка

Отметим, что перепутан порядок аргументов у инструкции add и что по окон- чании работы в edi отправляется ebx вместо eax

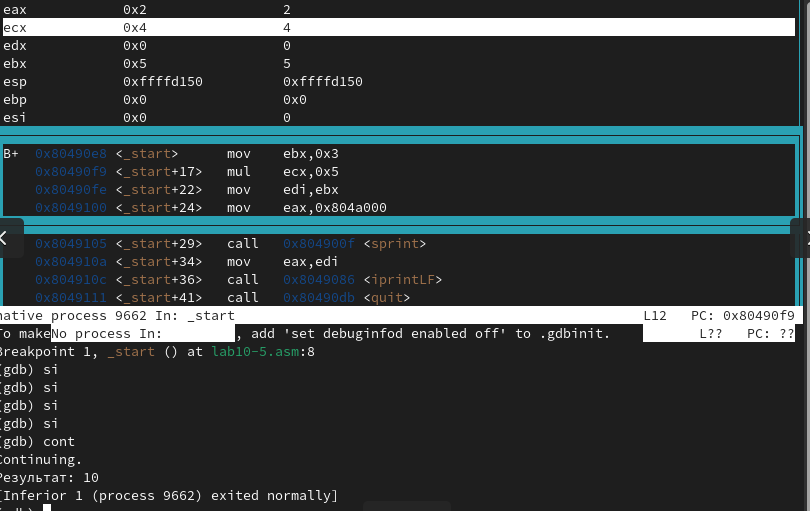


Рис. 4.20: код исправлен

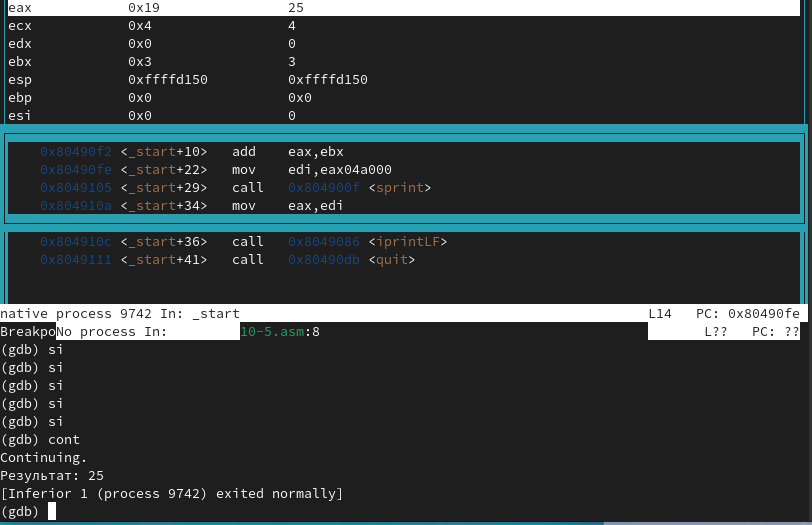


Рис. 4.21: проверка работы

# Выводы

Освоили работy с подпрограммами и отладчиком.

# Список литературы

* 1. [Расширенный ассемблер: NASM](https://www.opennet.ru/docs/RUS/nasm/)
  2. [MASM, TASM, FASM, NASM под Windows и Linux](https://habr.com/ru/post/326078/)