Отчёт по лабораторной работе №9

Простейший вариант

Бадалов Заури Эльвин оглы

Содержание

# 1 Цель работы

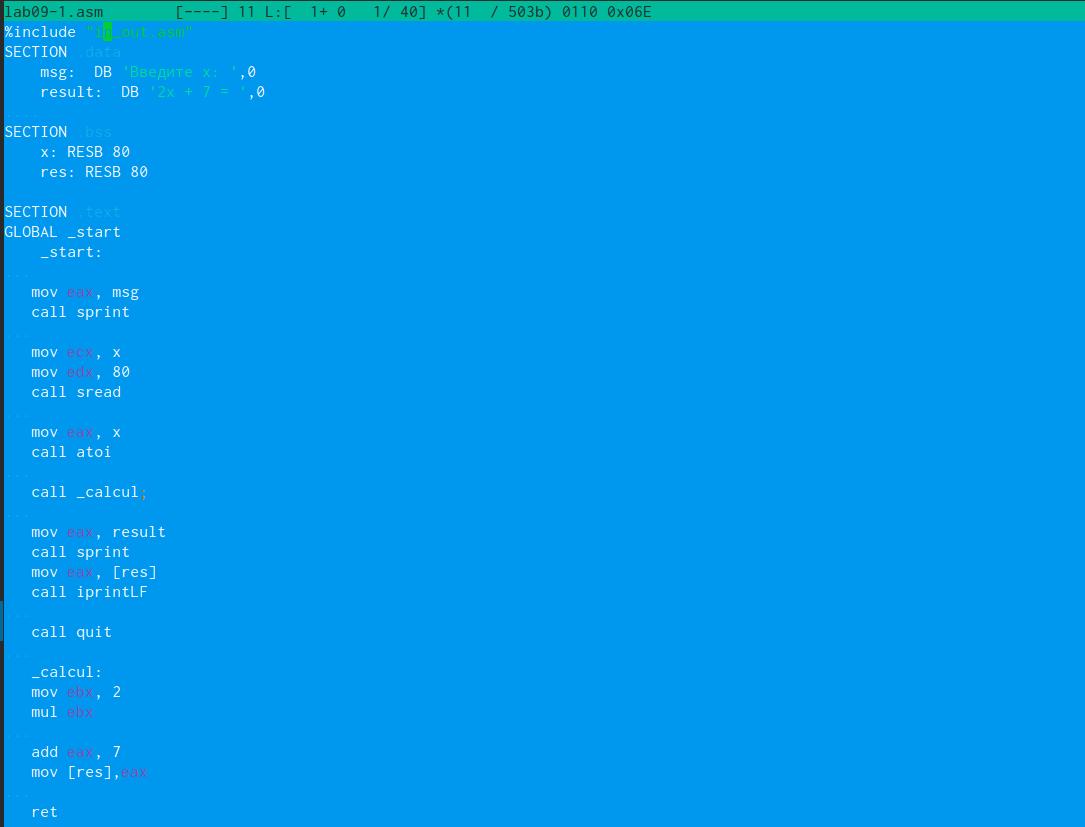
Освоить работу с подпрограммами и отладчиком gdb.

# 2 Задание

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

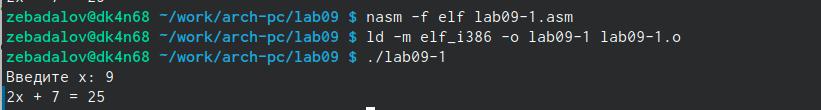
# 3 Выполнение лабораторной работы

Создаю рабочую директорию и файл. Записываю туда программу из листинга, исправив опечатки. (рис. ??).



Программа f(g(x))

Проверяю работу программы. (рис. ??).



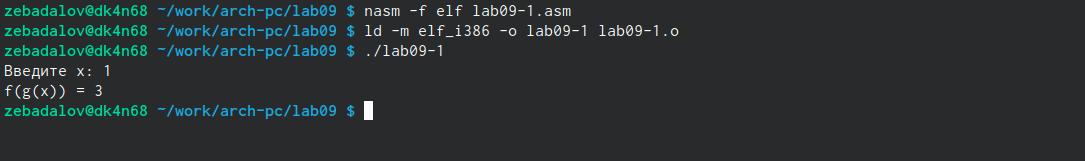
Запуск программы

Вношу текст программы, имитирующей сложную функцию. Функции назову \_calcul и \_subcalcul (рис. ??).



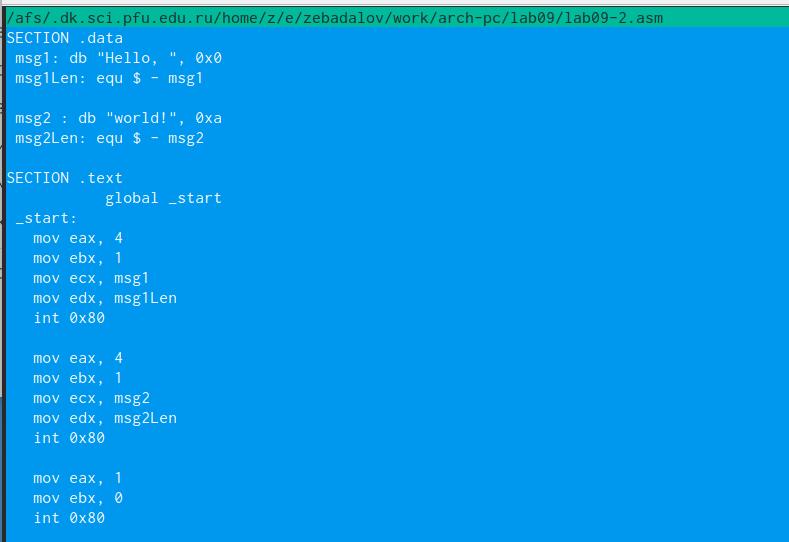
Программа с имитацией сложной функции

Проверяю работу программы. (рис. ??).



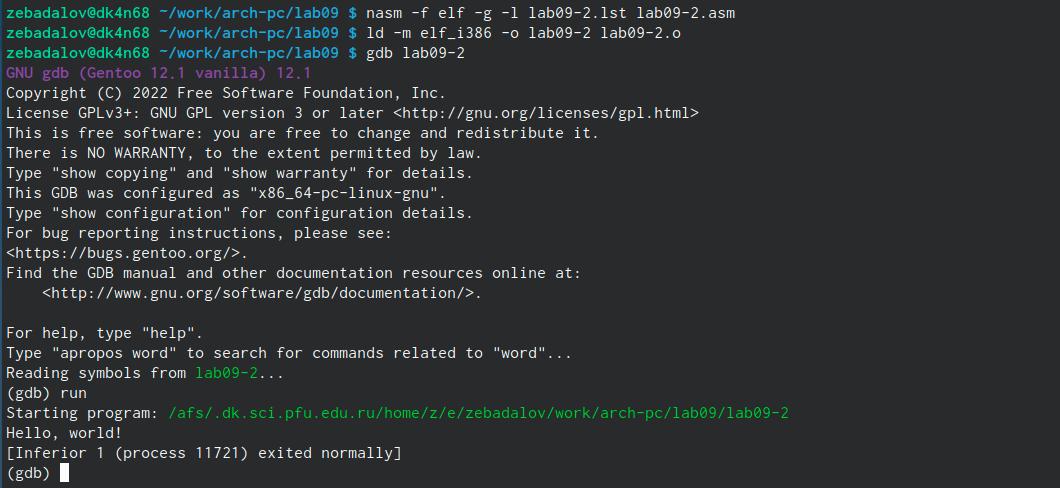
Запуск программы

Создаю файл lab09-2.asm и ввожу в него текст программы. (рис. ??).



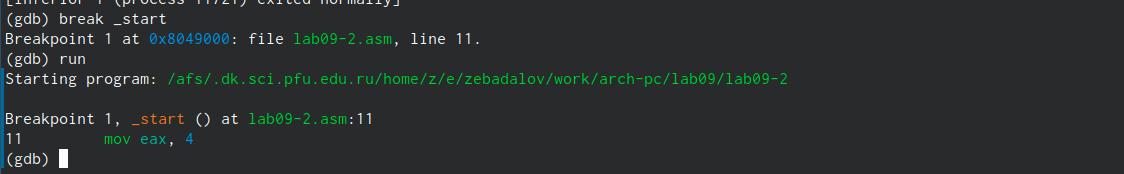
Текст программы файла lab09-2.asm

Транслирую файл с ключом -g. После загружаю исполняемый файл в отладчик gdb. Проверяю его работу запустив ее в отладчике gdb командой run. (рис. ??).



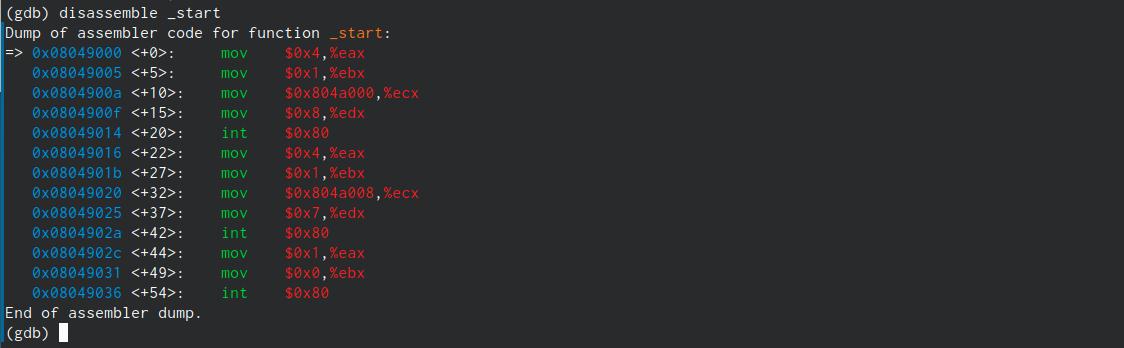
Работа файла в отладчике gdb

Устанавливаю брейкпоинт на метку \_start и запускаю ассемблерную программу. (рис. ??).



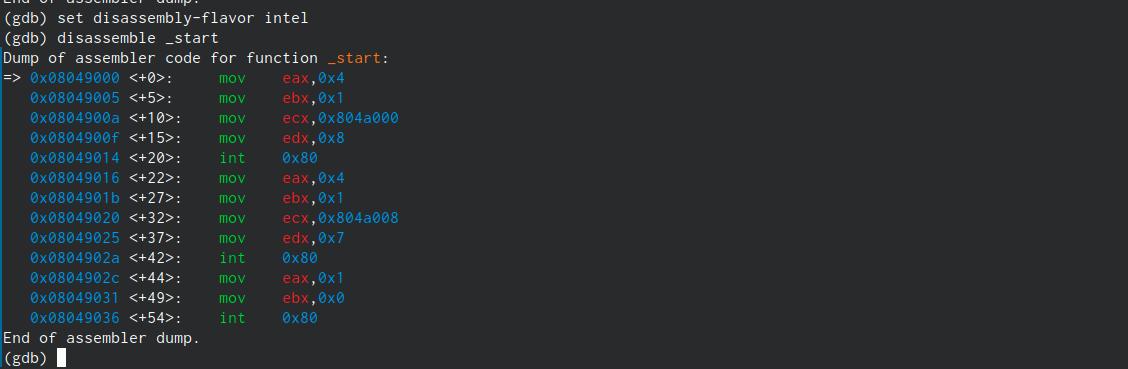
Брейкпоинт на метку \_start

Смотрю дисассимилированный код программы, использую команду disassemble, начиная с метки \_start. (рис. ??).



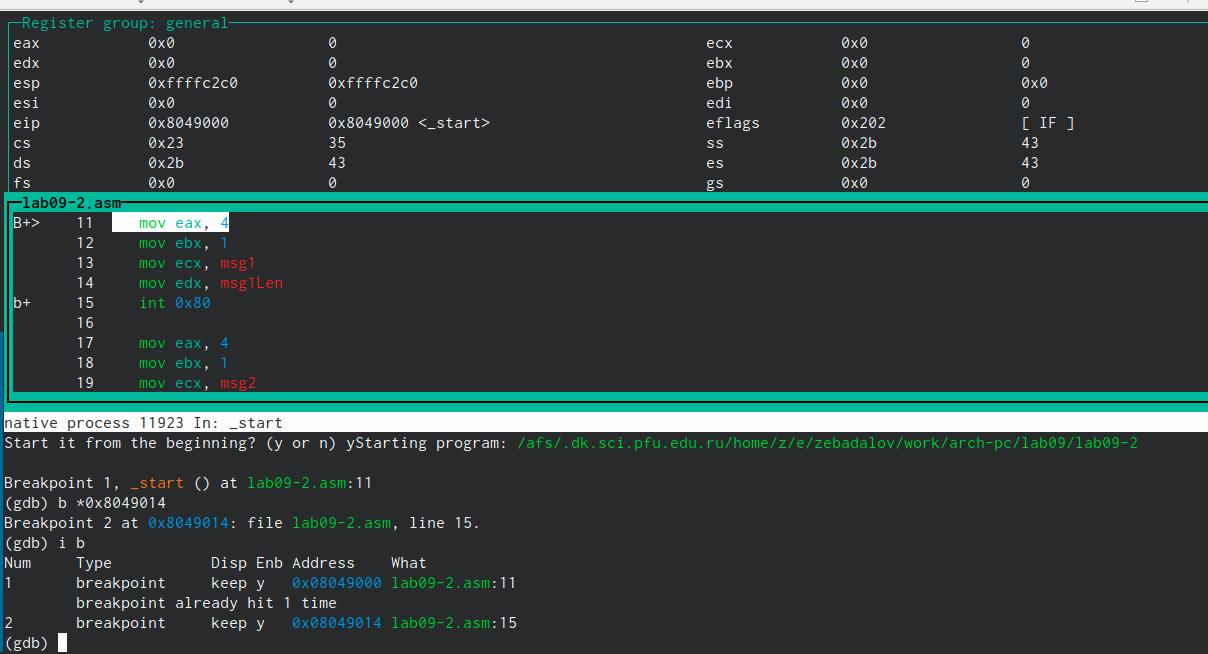
Дисассимилированный код программы

Переключаюсь на отображение команд с Intel’овским синтаксисом. (рис. ??).



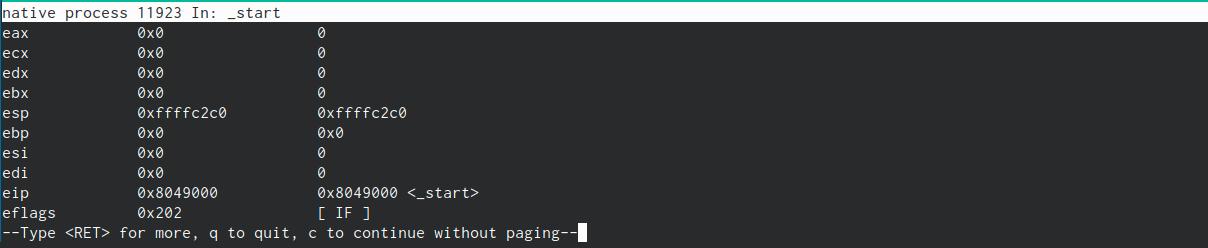
Отображение команд с Intel’овским синтаксисом

В представлении ATT в виде 16-ричного числа записаны первые аргументы всех комманд, а в представлении Intel так записываются адреса вторых аргументов. Включим режим псевдографики, при помощи которого отображается код программы и содержимое регистров. (рис. ??).



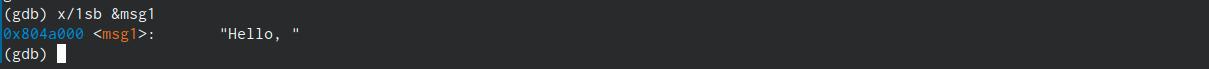
Режим псевдографики

Вывожу значения регистров, используя команду i r. (рис. ??).

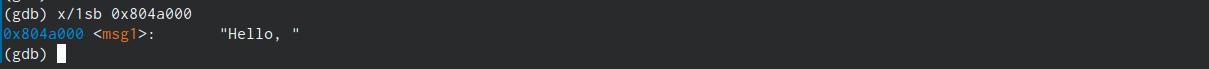


Значения регистров на псевдографике

В отладчике можно вывести текущие значения переменных. Сделать это можно, к примеру, по имени (рис. ??) или по адресу (рис. ??).

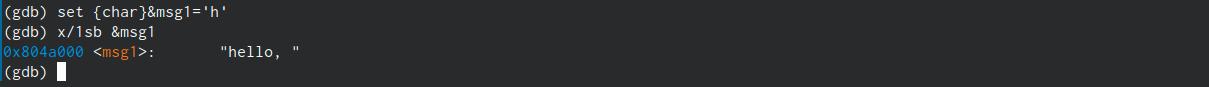


Получение значения переменной по имени



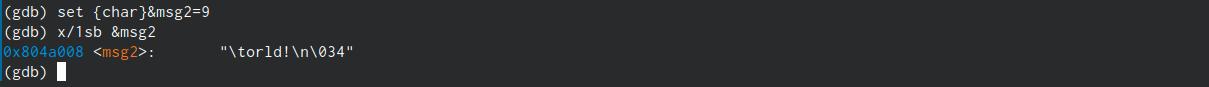
Получение значения переменной по адресу

Так же отладчик даёт менять значения переменных прямо во время выполнения программы. (рис. ??).



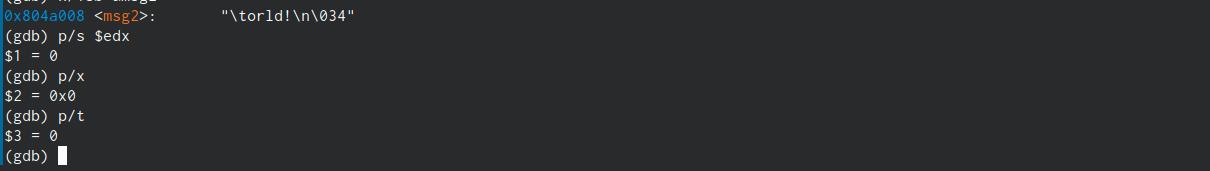
Замена значение переменной msg1

Заменю первый символ переменной msg2 на символ отступа. (рис. ??).



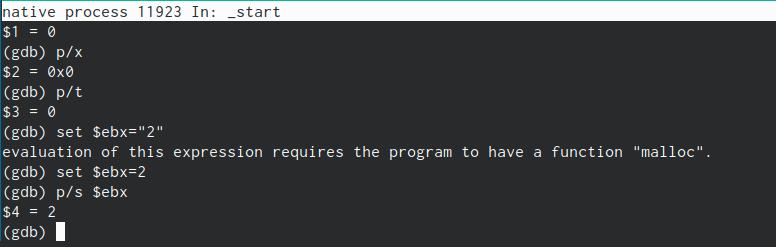
Замена символа переменной msg2

Выведу значение регистра edx в разных форматах: строчном, 16-ричном, двоичном. (рис. ??).



Вывод значения edx

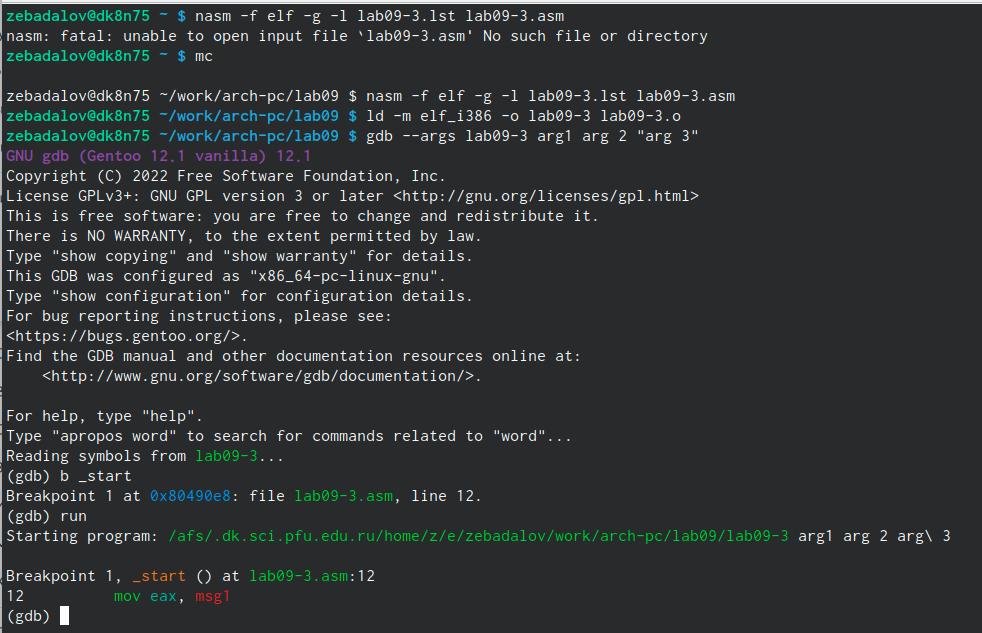
Регистрам тоже можно задавать значения. (рис. ??).



Присвоение значения регистру

Однако при попытке задать строчное значение, происходит ошибка. Завершу работу в gdb командами continue, она закончит выполнение программы, и exit, она завершит сеанс gdb.

Скопирую файл из лабораторной работы №9, переименую и создаю исполняемый файл. Открываю отладчик и задаю аргументы. Создаю точку останова на метке \_start и запускаю программу. (рис. ??).



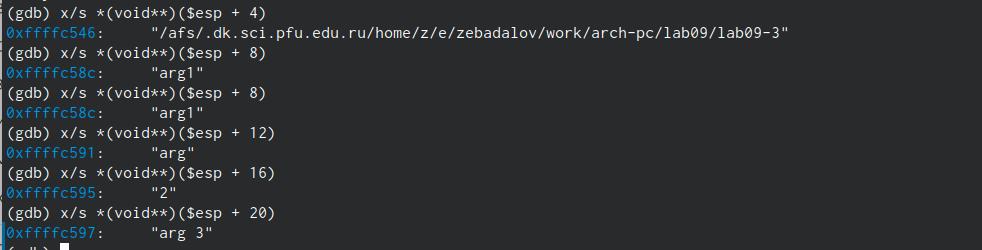
Запуск программы lab09-3.asm посредством gdb

Смотрю на содержимое того, что расположено по адресу, находящемуся в регисре esp. (рис. ??).

Содержимое регистра esp

Содержимое регистра esp

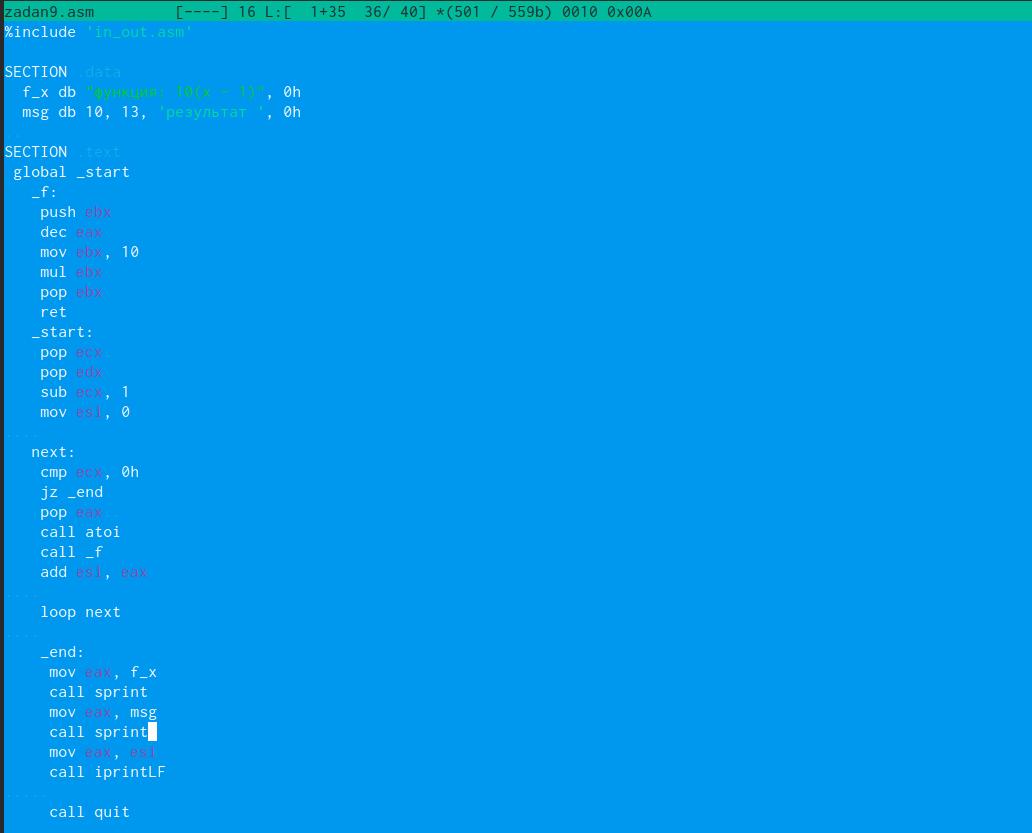
Далее смотрю на все остальные аргументы в стеке. Их адреса распологаются в 4 байтах друг от друга (именно столько занимает элемент стека). (рис. ??).



Остальные аргументы в стеке

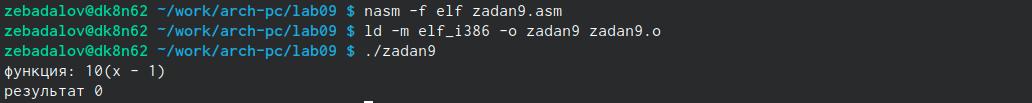
#Задания самостоятельной работы

Программа из лабораторной работы №8 с использованием подпрограмм. (рис. ??).



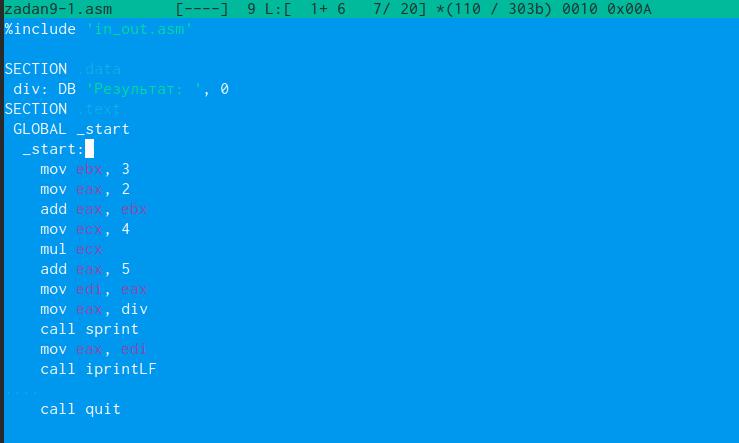
Программа файла zadan9

Проверяю результат выполнения работы программы. (рис. ??).



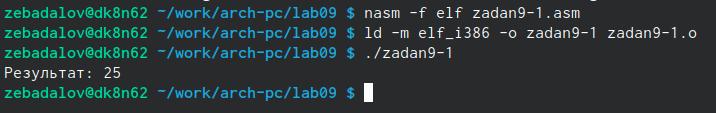
Запуск программы

Редактирую текст программы из листинга 9.3 в новом файле zadan9-1.asm после просмотра регистров для поиска ошибки. (рис. ??).



Исправленный текст программы

Проверяю работу программы. (рис. ??).



Запуск программы

# 4 Вывод

Я освоил работу с подпрограммами и отладчиком gdb.