Шаблон отчёта по лабораторной работе

9

Бембо Жозе Лумингу

Содержание

[1 Цель работы 1](#__RefHeading___1)

[2 Задание 1](#__RefHeading___2)

[3 Выполнение лабораторной работы 2](#__RefHeading___3)

[3.1 Реализация подпрограмм в NASM. 2](#__RefHeading___4)

[3.2 Отладка программам с помощью GDB. 3](#__RefHeading___5)

[3.3 Добавление точек останова. 6](#__RefHeading___6)

[3.4 Работа с данными программы в GDB. 7](#__RefHeading___7)

[3.5 Обработка аргументов командной строки в GDB. 10](#__RefHeading___8)

[3.6 Задания для самостоятельной работы. 11](#__RefHeading___16)

[4 Выводы 12](#__RefHeading___10)

[Список литературы 12](#__RefHeading___11)

# 1 Цель работы

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

# 2 Задание

Реализация подпрограмм в NASM.

Отладка программам с помощью GDB.

Добавление точек останова.

Работа с данными программы в GDB.

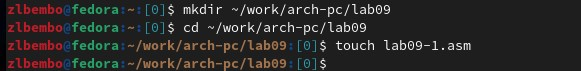
Обработка аргументов командной строки в GDB.

Задания для самостоятельной работы.

# 3 Выполнение лабораторной работы

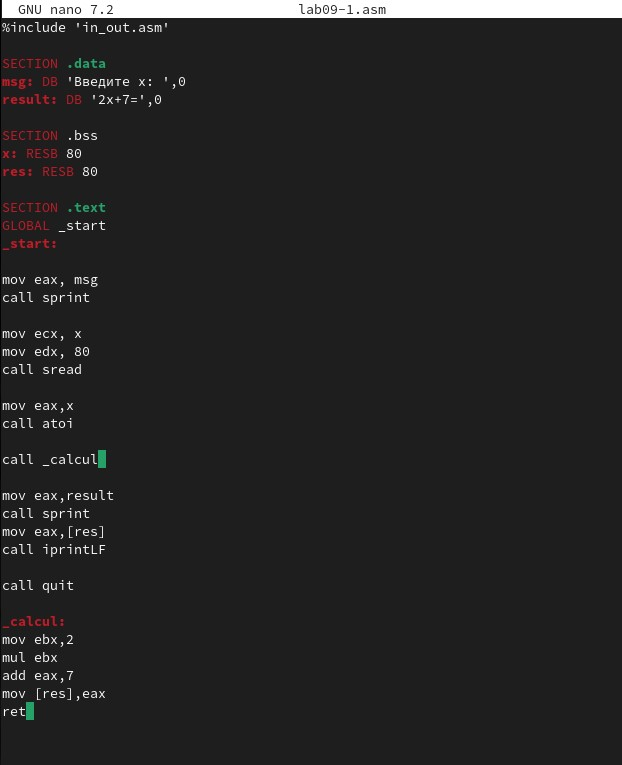
## 3.1 Реализация подпрограмм в NASM.

Создаю каталог для выполнения лабораторной работы № 9, перехожу в него и создаю файл lab09-1.asm. (рис. [??]).



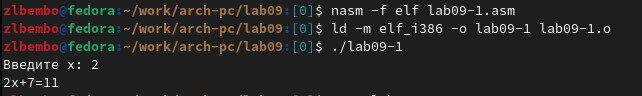
создание файлов для лабораторной работы

Ввожу в файл lab09-1.asm текст программы с использованием подпрограммы из листинга 9.1. (рис. [??]).



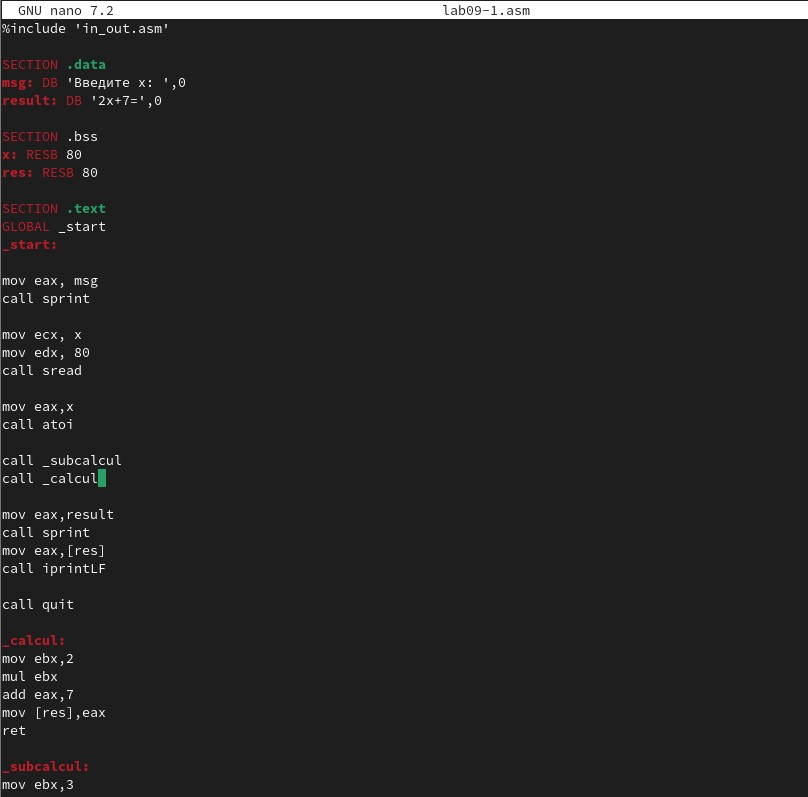
ввод текста программы

Создаю исполняемый файл и проверяю его работу. (рис. [??]).



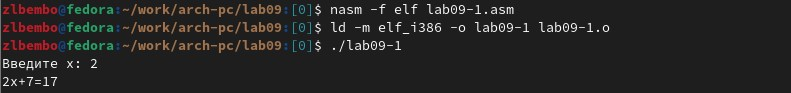
запуск исполняемого файла

Изменяю 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.(рис. [??]).



изменение текста программы

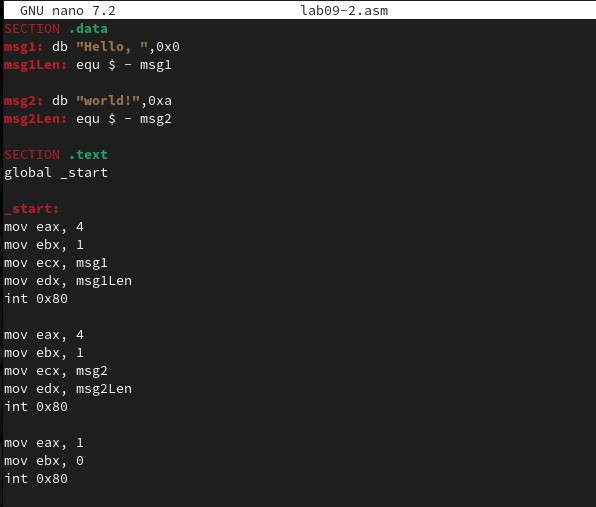
Создаю исполняемый файл и проверяю его работу. (рис. [??]).



запуск исполняемого файла

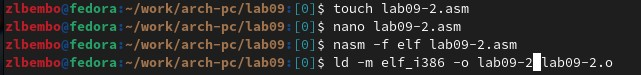
## 3.2 Отладка программам с помощью GDB.

На этом шаге мы создали файл lab09-2.asm с текстом программы из ли- стинга 9.2. (рис. [??]).



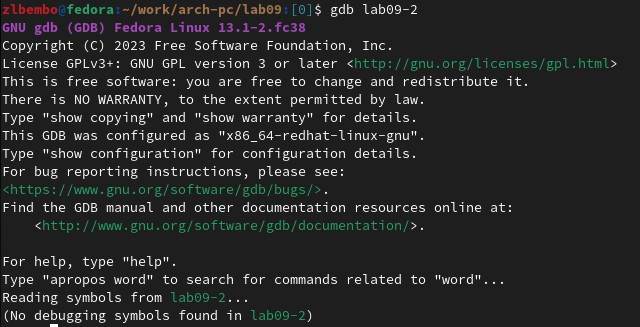
ввод текста программы

Получаю исполняемый файл для работы с GDB с ключом ‘-g’. (рис. [??]).



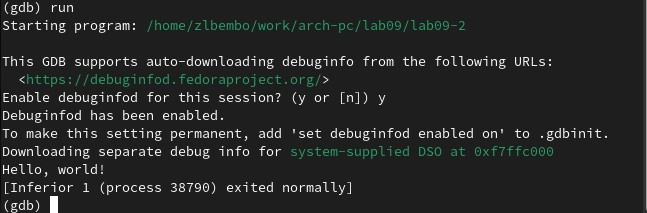
получение исполняемого файла

Загружаю исполняемый файл в отладчик gdb.(рис. [??]).



загрузка исполняемого файла в отладчике

Проверяю работу программы, запустив ее в оболочке GDB с помощью команды run. (рис. [??]).



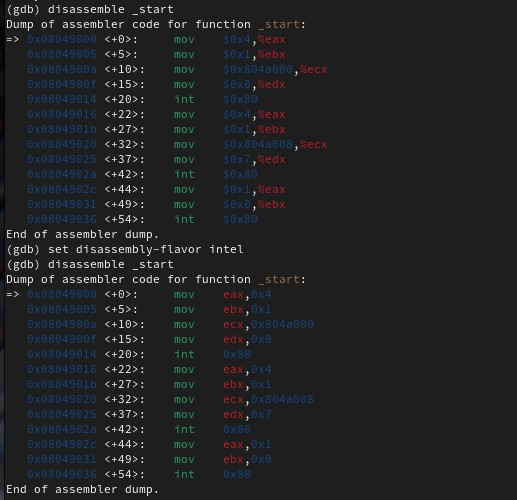
проверка работы файла с помощью команды run

Для более подробного анализа программы устанавливаю брейкпоинт на метку \_start и запускаю её.(рис. [??]).



установка брейкпоинта и запуск программы

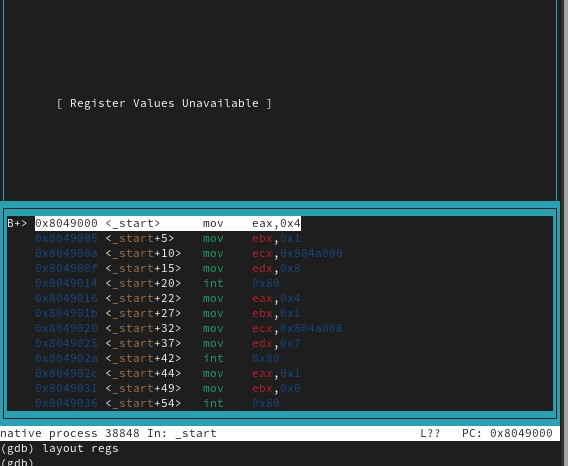
Просматриваю дисассимилированный код программы с помощью команды disassemble, начиная с метки \_start, и переключаюсь на отображение команд с синтаксисом Intel, введя команду set disassembly-flavor intel.(рис. [??]).



использование команд disassemble и set disassembly-flavor intel

Разница в синтаксисе между AT&T и INTEL заключается в том, что AT&T использует синтаксис mov $0x4,%eax, который популярен среди пользова- телей Linux, с другой стороны, INTEL использует синтаксис mov eax,0x4 , который является популярен среди пользователей Windows.

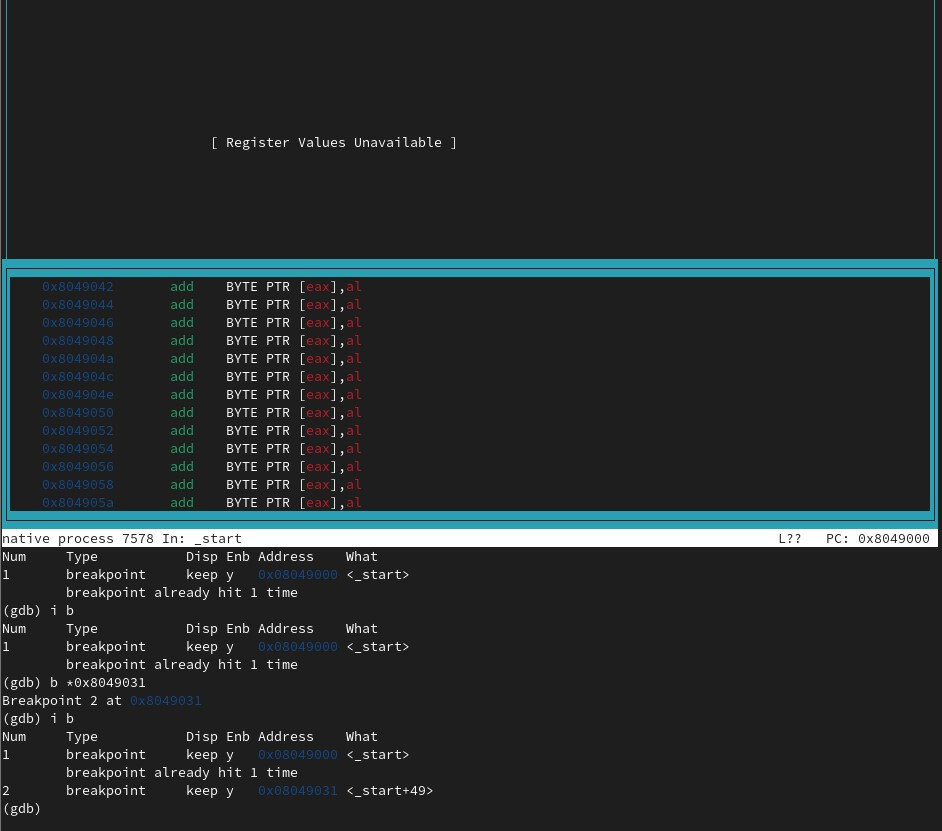
Включаю режим псевдографики для более удобного анализа программы с помощью команд layout asm и layout regs.(рис. [??]).



включение режима псевдографики

## 3.3 Добавление точек останова.

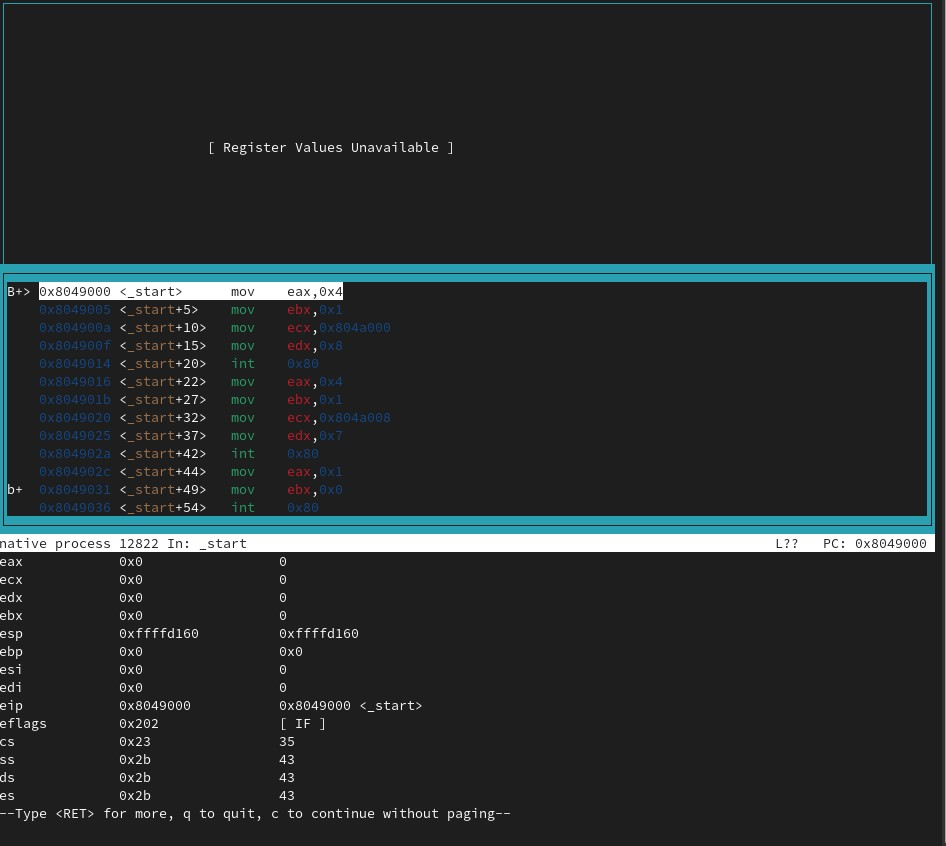
Проверяю, что точка останова по имени метки \_start установлена с помощью команды info breakpoints и устанавливаю еще одну точку останова по адресу инструкции mov ebx,0x0. Просматриваю информацию о всех установленных точках останова.(рис. [??]).



установление точек остановка

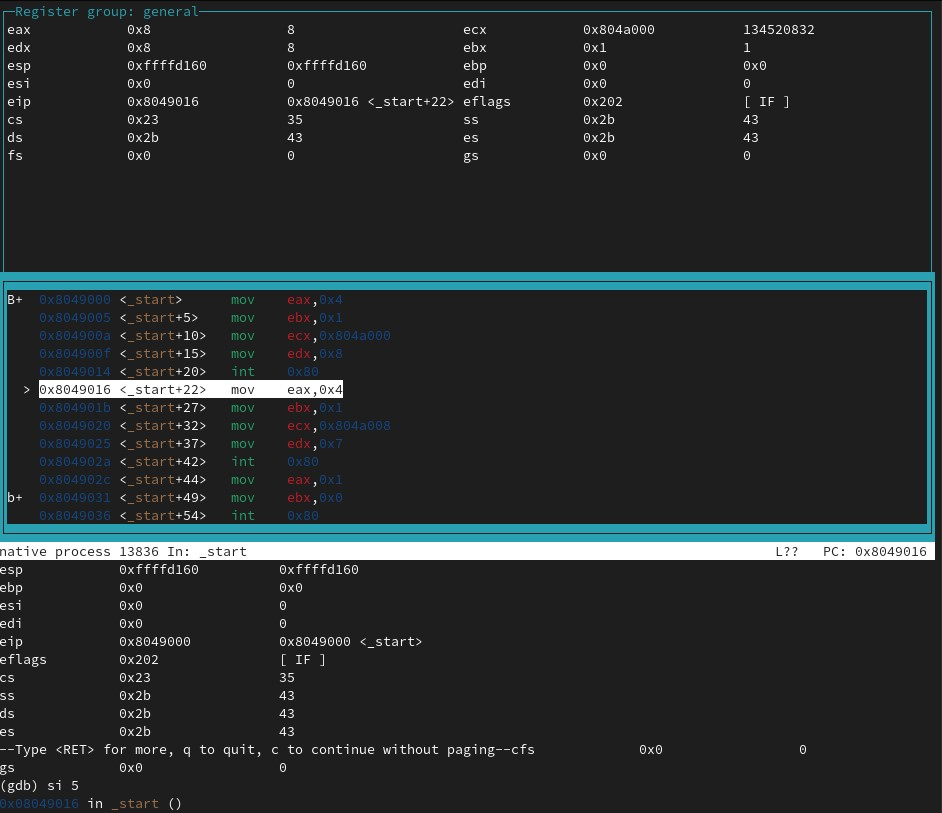
## 3.4 Работа с данными программы в GDB.

Выполняю 5 инструкций с помощью команды stepi и слежу за изменением значений регистров. (рис. [??]).



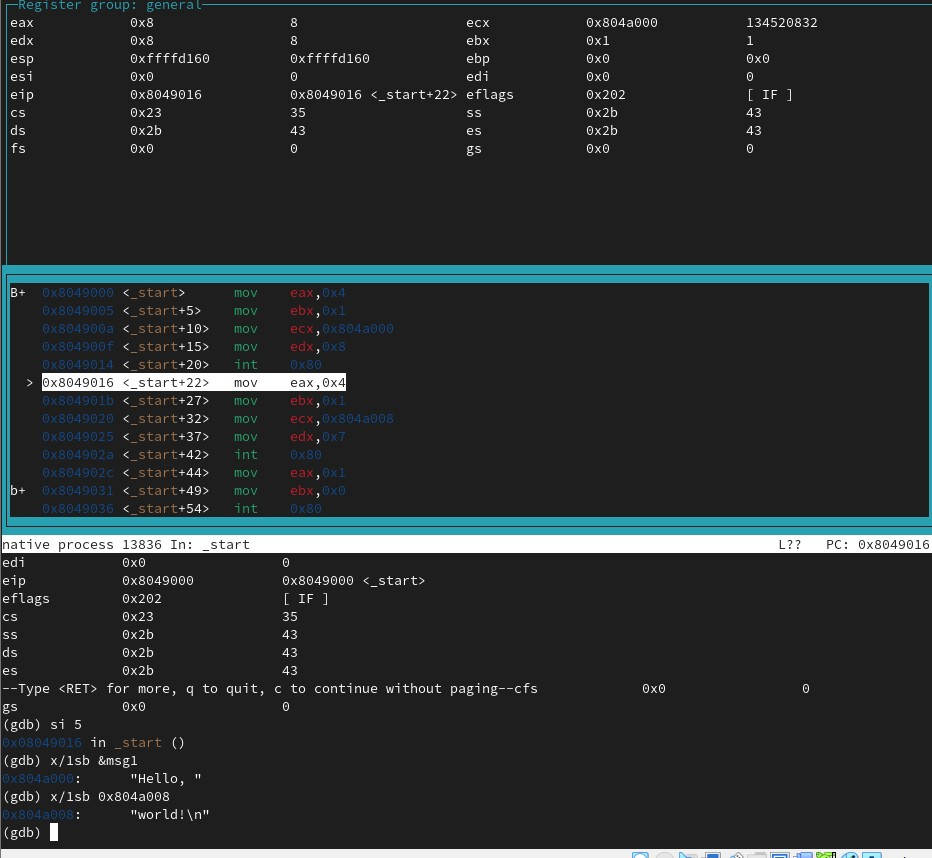
до использования команды stepi

(рис. [??]).



после использования команды stepi

Просматриваю значение переменной msg1 по имени с помощью команды x/1sb &msg1 и значение переменной msg2 по ее адресу.(рис. [??]).



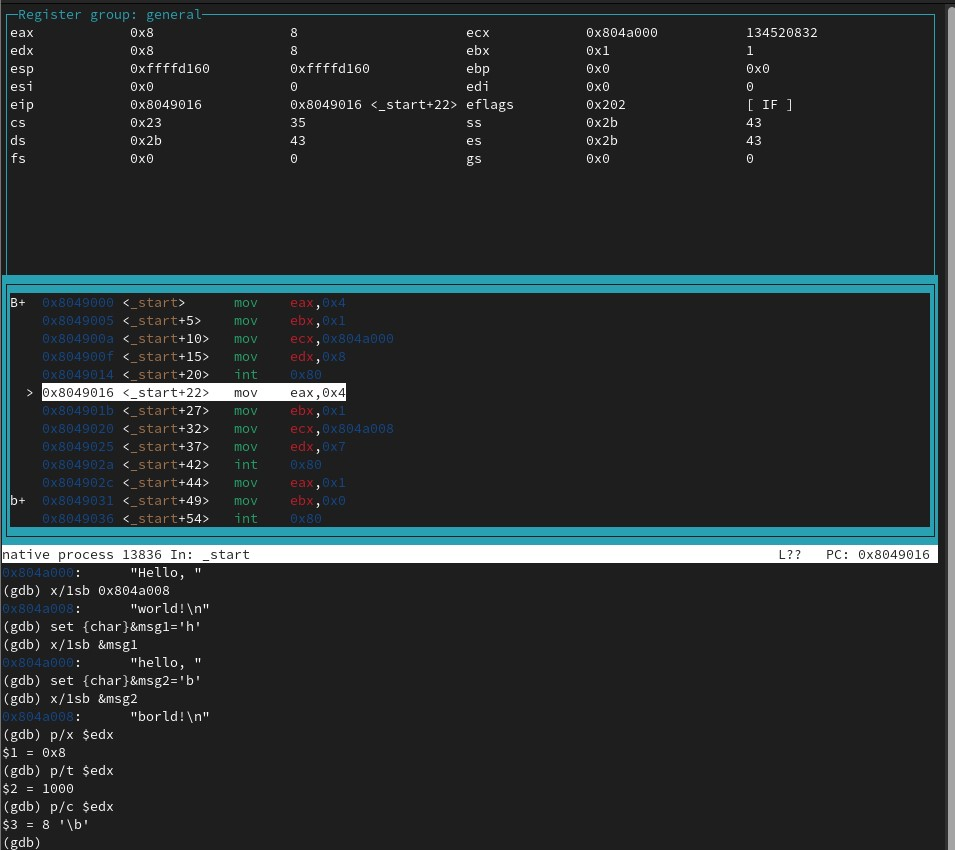
просмотр значений переменных

С помощью команды set изменяю первый символ переменной msg1 и заменяю первый символ в переменной msg2.(рис. [??]).



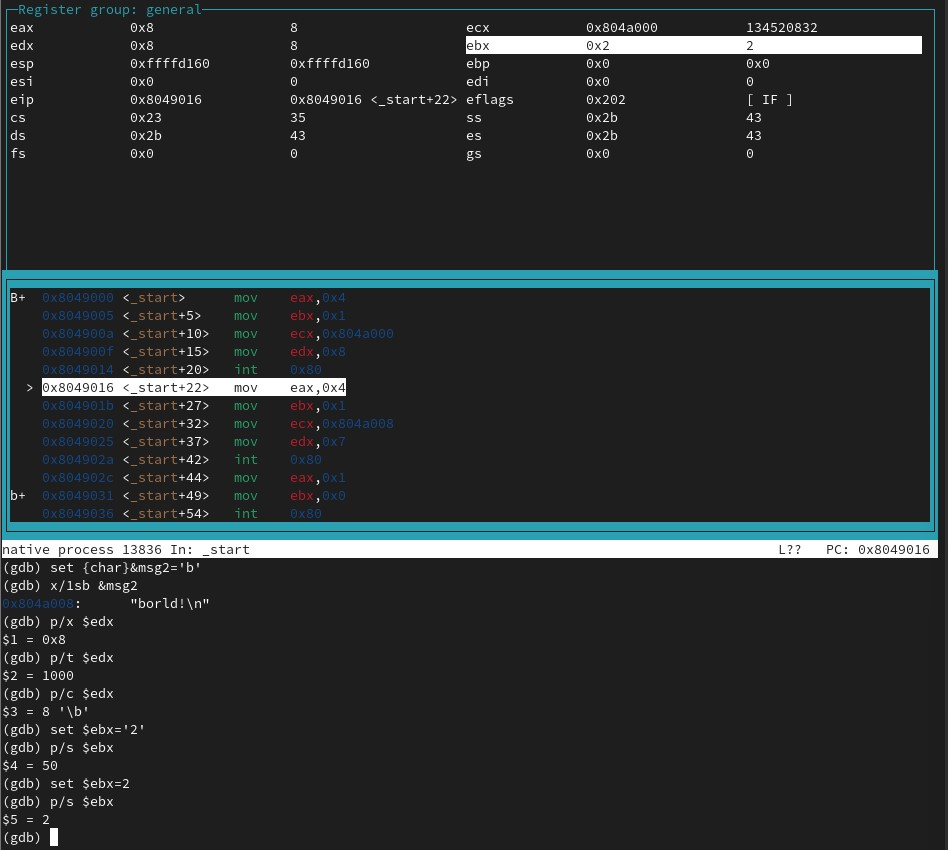
использованиe команды set

Вывожу в шестнадцатеричном формате, в двоичном формате и в символьном виде соответственно значение регистра edx с помощью команды print p/F $val. (рис. [??]).



вывод значения регистра

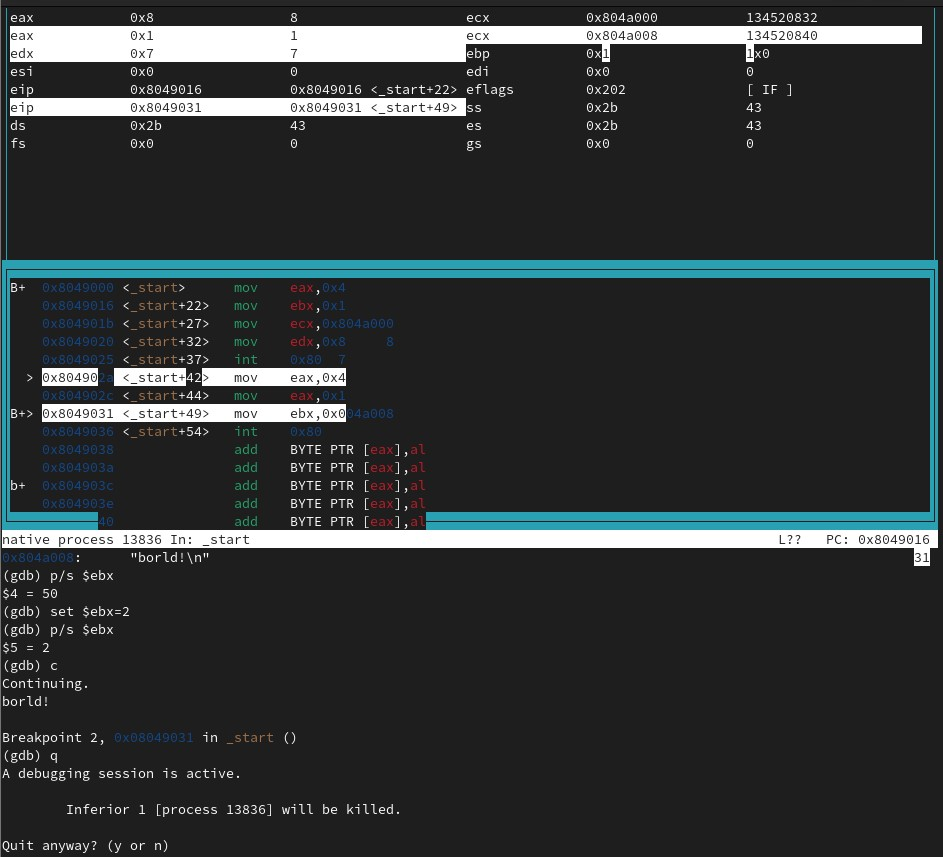
С помощью команды set изменяю значение регистра ebx в соответствии с заданием. (рис. [??]).



использование команды set для изменения значения регистра

Разница вывода команд p/s $ebx отличается тем, что в первом случае мы переводим символ в его строковый вид, а во втором случае число в строковом виде не изменяется.

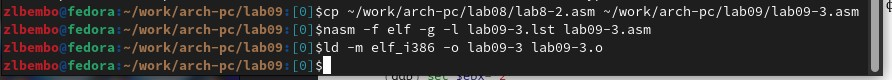
Завершаю выполнение программы с помощью команды continue и выхожу из GDB с помощью команды quit.(рис. [??]).



завершение работы

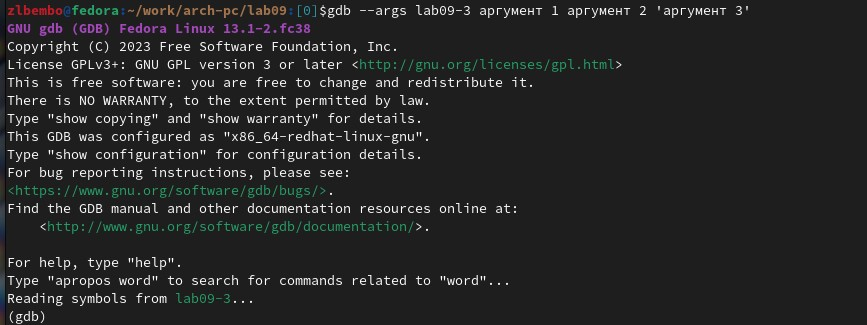
## 3.5 Обработка аргументов командной строки в GDB.

Копирую файл lab8-2.asm с программой из листинга 8.2 в файл с именем lab09-3.asm и создаю исполняемый файл. (рис. [??]).



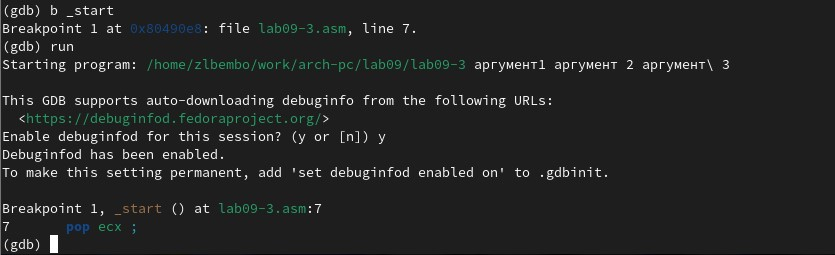
создание файла

Загружаю исполняемый файл в отладчик gdb, указывая необходимые аргументы с использованием ключа –args.(рис. [??]).



загрузка исполняемого файла в отладчике

Устанавливаю точку останова перед первой инструкцией в программе и запускаю ее.(рис. [??]).



установление точек остановка

Посматриваю вершину стека и позиции стека по их адресам. (рис. [??]).

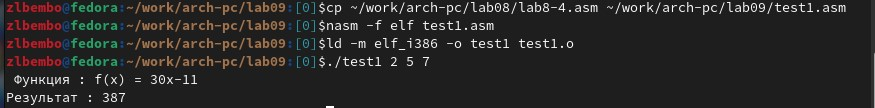


просмотр значений и введение в стек

## 3.6 Задания для самостоятельной работы.

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

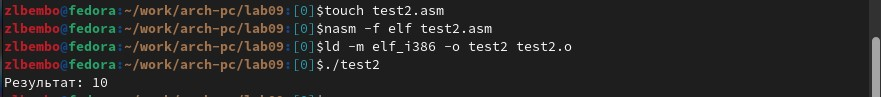
Запускаю код и проверяю, что она работает корректно. (рис. [??]).



запуск программы

Ввожу в файл task1.asm текст программы из листинга 9.3

При корректной работе программы должно выводится “25”. Создаю исполняемый файл и запускаю его.(рис. [??]).



запуск программы

# 4 Выводы

Во время выполнения данной лабораторной работы я приобрела навыки написания программ с использованием подпрограмм и ознакомилась с методами отладки при помощи GDB и его основными возможностями.

# Список литературы

GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.

GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/.

Midnight Commander Development Center. — 2021. — URL: https://midnight-commander. org/.

NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.

Newham C. Learning the bash Shell: Unix Shell Programming. — O’Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learning- bash-Shell-Programming-Nutshell/dp/0596009658.

Robbins A. Bash Pocket Reference. — O’Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.

The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.

Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.

Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.

Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс,

Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.

Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/.

Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ- Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.

Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm\_unix.

Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).

Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,

— 1120 с. — (Классика Computer Science).