Лабораторная работа №6

Дисциплина: Архитектура компьютера

Савостин Олег

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

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

# 2 Задание

1. Реализация подпрограмм в NASM
2. Отладка программам с помощью GDB
3. Выполнение заданий для самостоятельной работы

# 3 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. Можно выделить следующие типы ошибок: • синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; • семантические ошибки — являются логическими и приводят к тому, что программа запускается, отрабатывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают пре- рывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль). Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить доволь- но трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга. Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы. Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново.

Наиболее часто применяют следующие методы отладки: • создание точек контроля значений на входе и выходе участка программы (например, вывод промежуточных значений на экран — так называемые диагностические сообще- ния); • использование специальных программ-отладчиков. Отладчики позволяют управлять ходом выполнения программы, контролировать и из- менять данные. Это помогает быстрее найти место ошибки в программе и ускорить её исправление. Наиболее популярные способы работы с отладчиком — это использование точек останова и выполнение программы по шагам. Пошаговое выполнение — это выполнение программы с остановкой после каждой строчки, чтобы программист мог проверить значения переменных и выполнить другие действия. Точки останова — это специально отмеченные места в программе, в которых программа- отладчик приостанавливает выполнение программы и ждёт команд. Наиболее популярные виды точек останова

GDB (GNU Debugger — отладчик проекта GNU) [1] работает на многих UNIX-подобных системах и умеет производить отладку многих языков программирования. GDB предлагает обширные средства для слежения и контроля за выполнением компьютерных программ. От- ладчик не содержит собственного графического пользовательского интерфейса и использует стандартный текстовый интерфейс консоли. Однако для GDB существует несколько сторон- них графических надстроек, а кроме того, некоторые интегрированные среды разработки используют его в качестве базовой подсистемы отладки

Синтаксис команды для запуска отладчика имеет следующий вид: gdb [опции] [имя\_файла | ID процесса]

Если есть файл с исходным текстом программы, а в исполняемый файл включена информа- ция о номерах строк исходного кода, то программу можно отлаживать, работая в отладчике непосредственно с её исходным текстом. Чтобы программу можно было отлаживать на уровне строк исходного кода, она должна быть откомпилирована с ключом -g. Посмотреть дизассемблированный код программы можно с помощью команды disassemble : (gdb) disassemble \_start

Установить точку останова можно командой break (кратко b). Типичный аргумент этой команды — место установки. Его можно задать как имя метки или как адрес. Чтобы не было путаницы с номерами, перед адресом ставится «звёздочка»: (gdb) break \* (gdb) b Информацию о всех установленных точках останова можно вывести командой info (крат- ко i): (gdb) info breakpoints (gdb) i b Как уже упоминалось, отладчик может показывать содержимое ячеек памяти и регистров, а при необходимости позволяет вручную изменять значения регистров и переменных. Посмотреть содержимое регистров можно с помощью команды info registers (или i r): (gdb) info registers

Подпрограмма — это, как правило, функционально законченный участок кода, который можно многократно вызывать из разных мест программы. В отличие от простых переходов из подпрограмм существует возврат на команду, следующую за вызовом

# 4 Выполнение лабораторной работы

## 4.1 Реализация подпpограмм в NASM

Создаю нужные файлы и вставляю в .asm файл текст из листинга (рис. 1).(рис. 2).

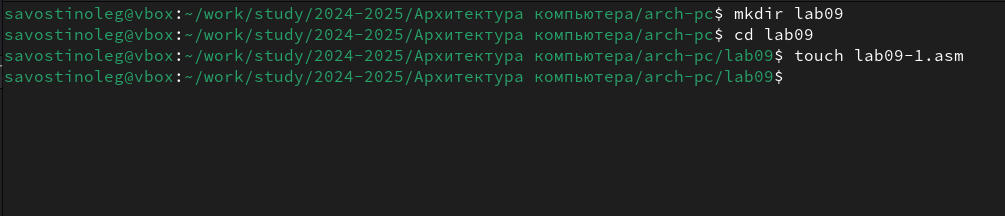


Рис. 1: Создание файлов

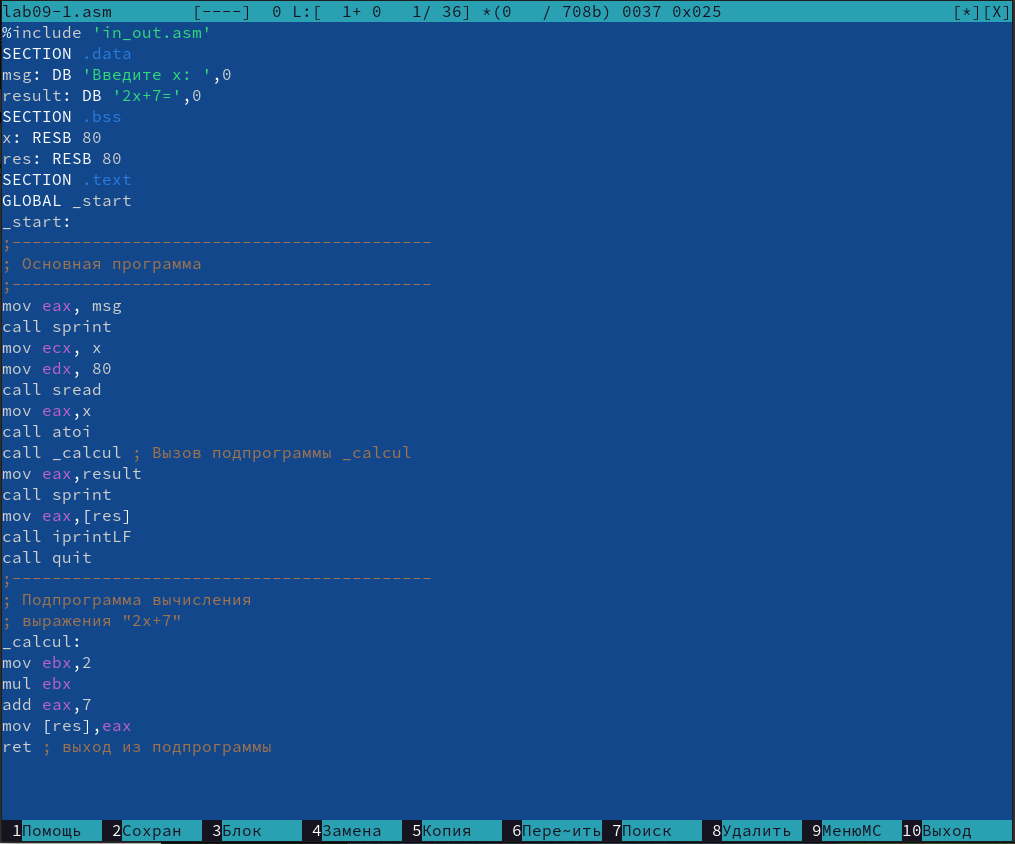


Рис. 2: Листинг 9.1

Проверяю его на работу и создаю исполняемый файл (рис. 3).

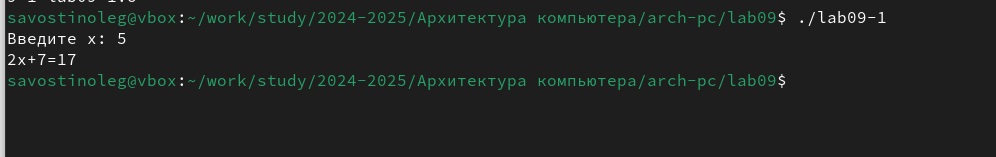


Рис. 3: Исполняемый файл Листинга 9.1

Изменяю текст файла Листинга (рис. 4), чтобы он имел выводил сложную функцию (рис. 5).

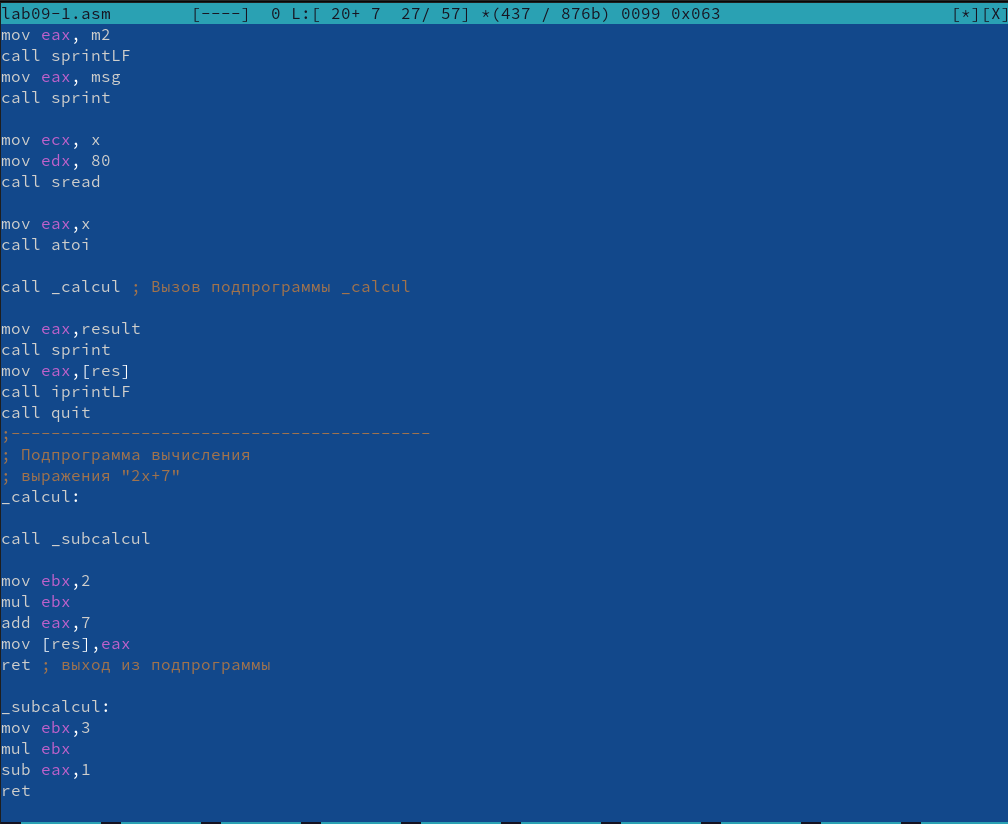


Рис. 4: Измененный Листинг

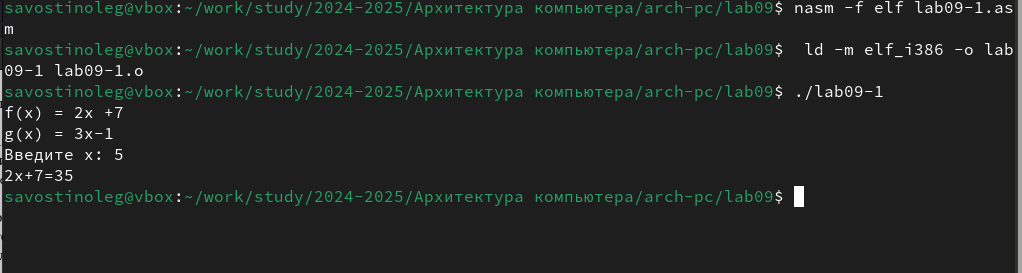


Рис. 5: Запуск исполняемого файла

## 4.2 Отладка программам с помощью GDB

Создаю файл и вставляю в него Листинг 9.2(рис. 6).

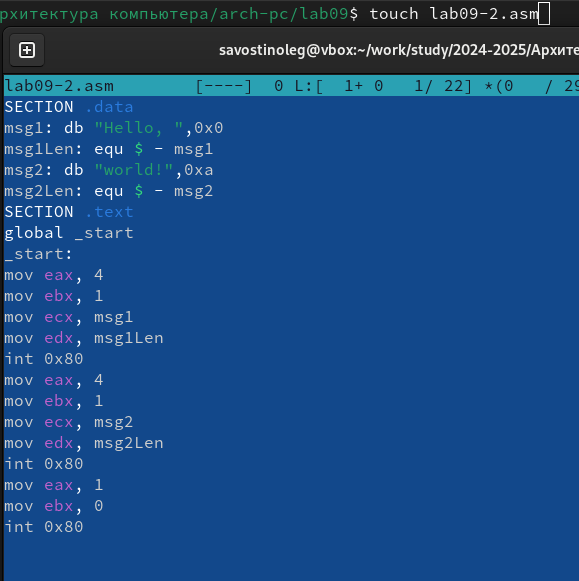


Рис. 6: Листинг 9.2

Создаю исполняемый файл и .lst (рис. 7) и запускаю gdb (рис. 8). Ставлю брекпоинт на \_start.

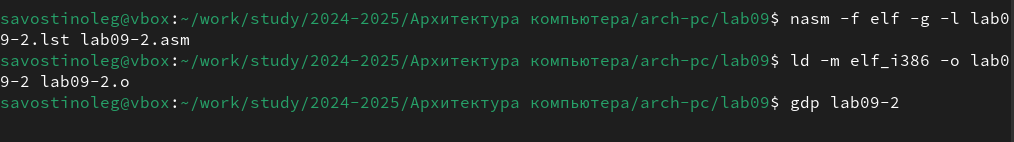


Рис. 7: Создание исполняемых файлов

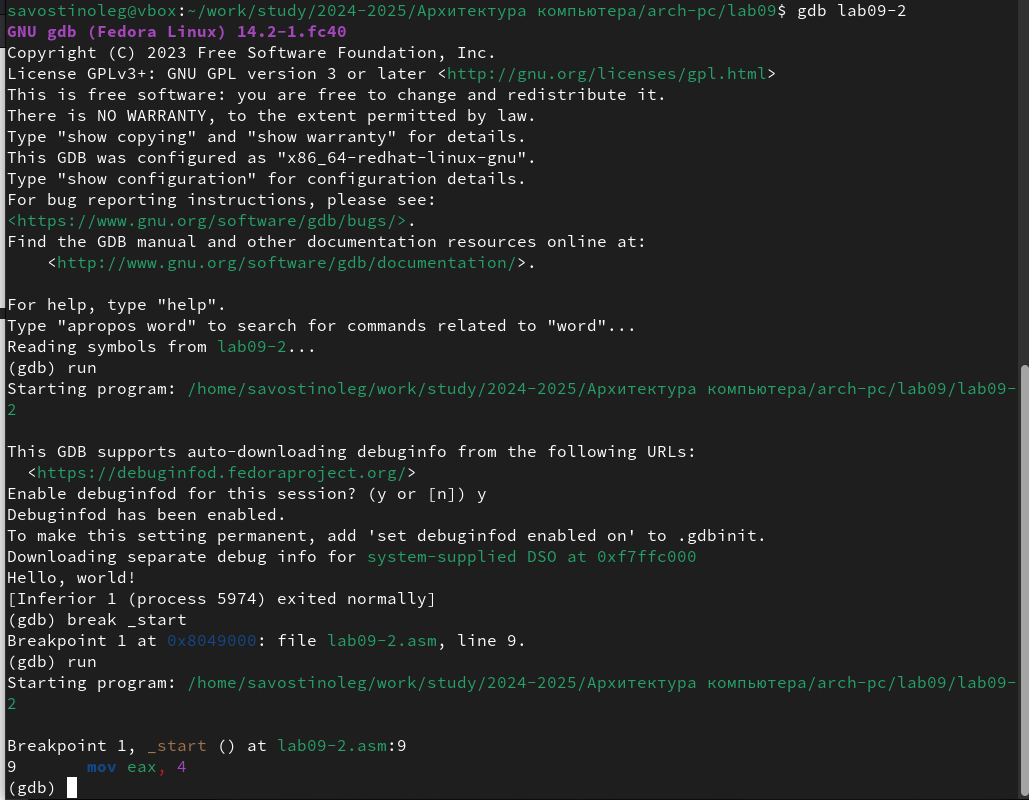


Рис. 8: Использование gdb

Начиная с метки, просматриваю дисассимплированный код (рис. 9) и переключаюсь на intel’овское отображение синтаксиса. Отличие заключается в командах, в диссамилированном отображении в командах используют % $, а в Intel эти символы не отображены, тем самым так удобнее.

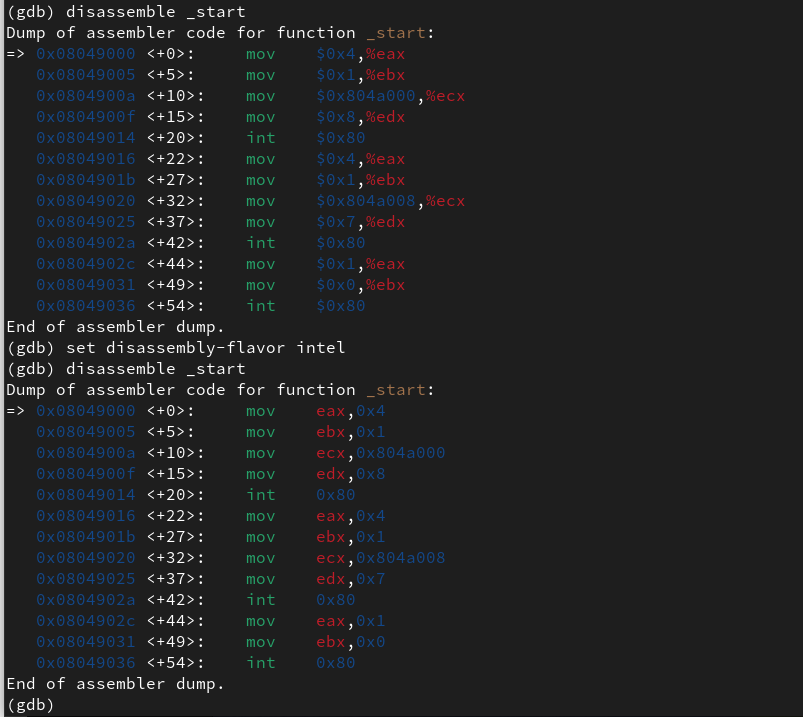


Рис. 9: Disassembly Intel

Использую layout asm (рис. 10) layout regs (рис. 11).

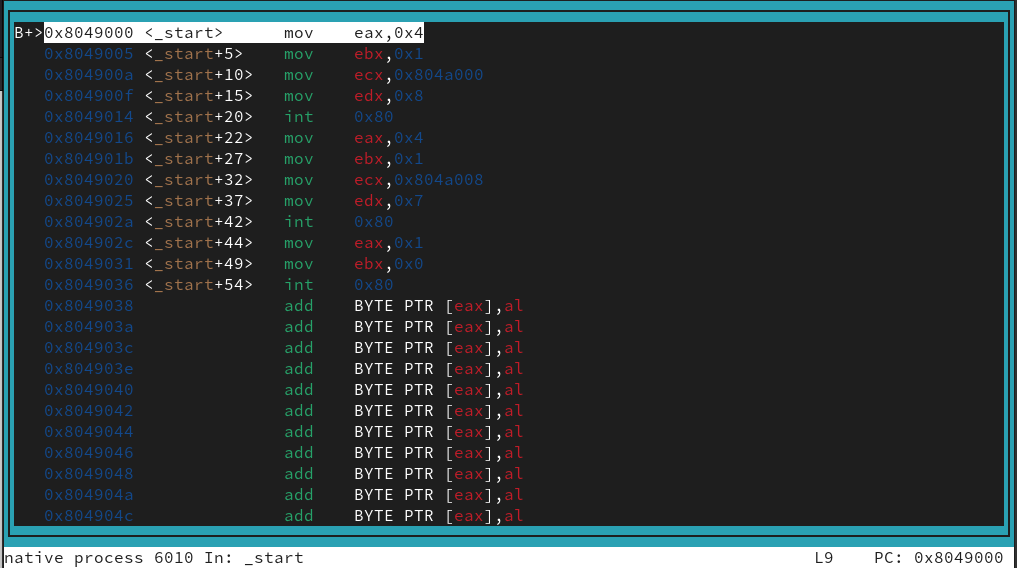


Рис. 10: layout asm

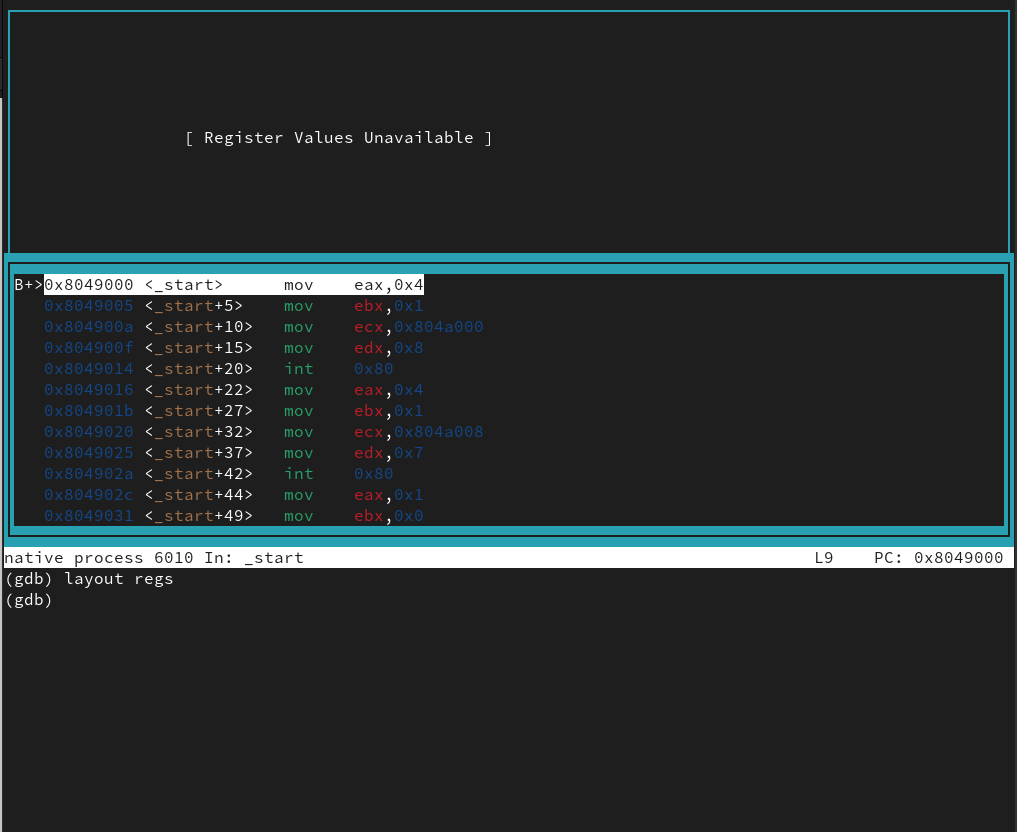


Рис. 11: layout regs

Проверяю на наличие меток (рис. 12). Добавляю одну метку(рис. 13).

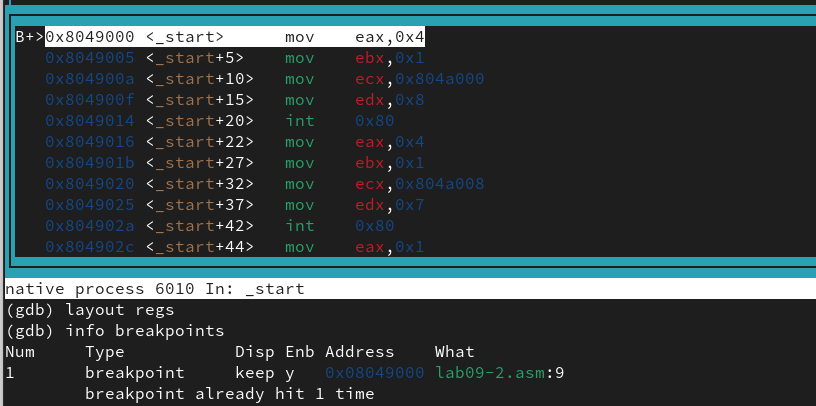


Рис. 12: Проверка на наличие меток

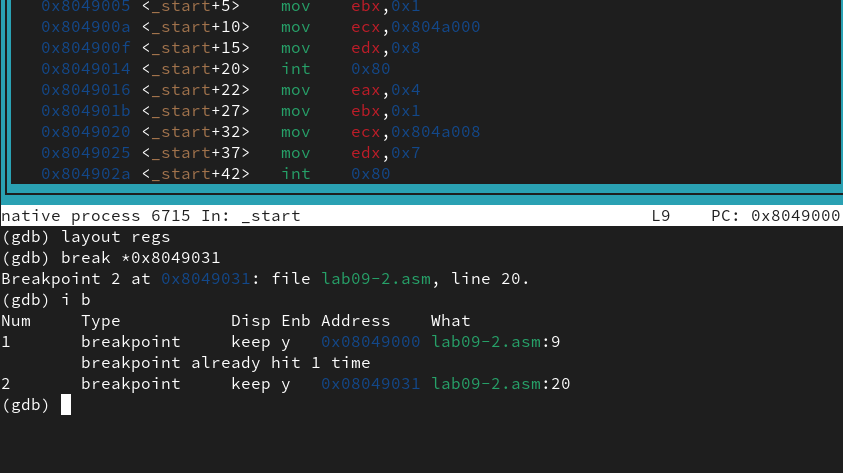


Рис. 13: Создание метки

С помощью команды si я посмотрел регистры (рис. 14).

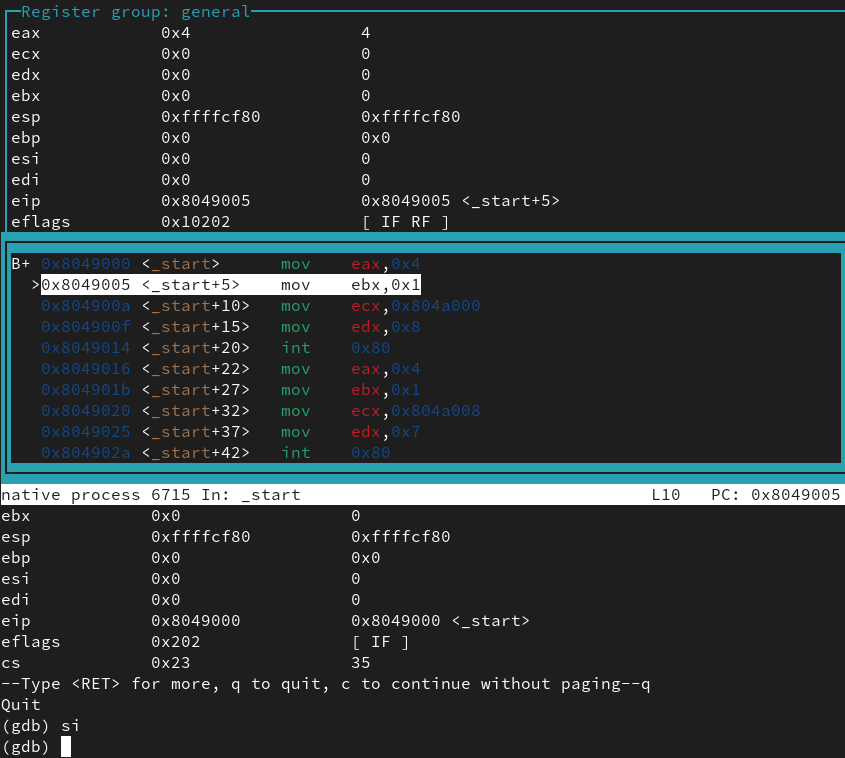


Рис. 14: Регистры si

Просматриваю значения msg1 msg2 и изменяю их(рис. 15).

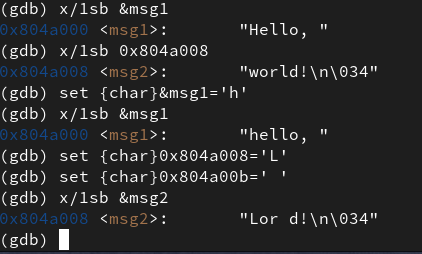


Рис. 15: Значения и изменения msg1 msg2

Изменяю значение регистра ebx. Выводятся два разных значения так как в первом случае вводится значение 2, а во второй раз регистр равен 2. (рис. 16).

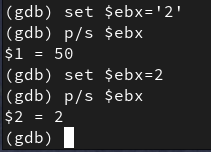


Рис. 16: Изменения значения регистра

Проверка значений регистров (рис. 17).

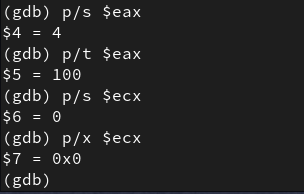


Рис. 17: Значения регистров

Выхожу из программы(рис. 18).

Рис. 18: Выход

Рис. 18: Выход

Теперь, я скопировал файл lab8-2.asm и переименовал его в lab09-3.asm. Добавляю метку и запускаю файл(рис. 19).

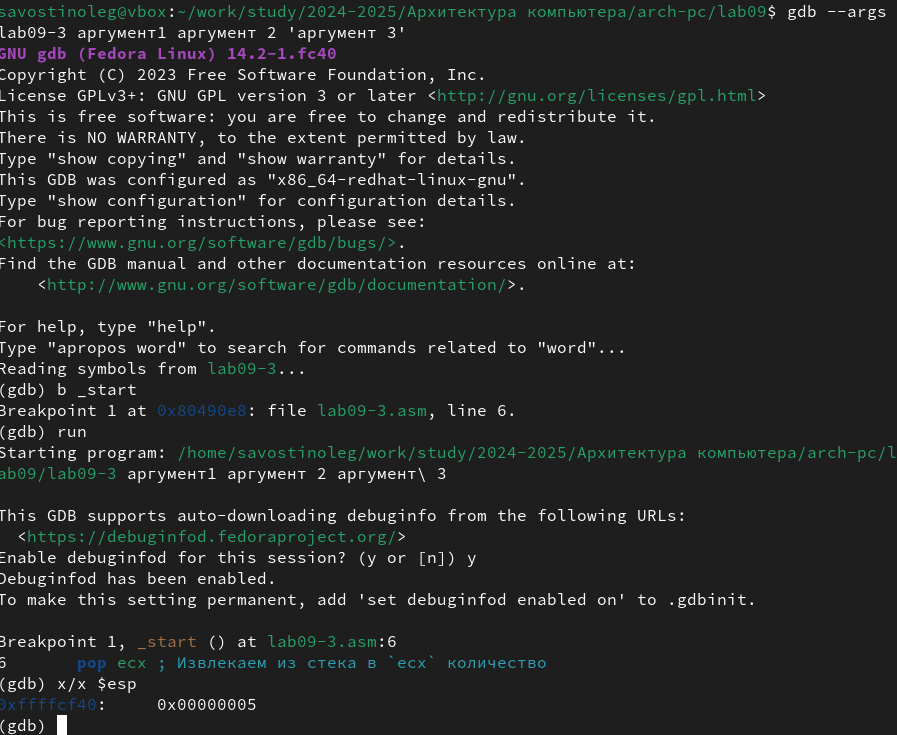


Рис. 19: Файл из 8 Лабораторной работы

Проверяю адрес вершины стека. Там хранится 5 элементов. По первому адресу находится адрес а в остальных хранятся элементы. Элементы расположены с интервалом в 4 единиицы. Стек может хранить до 4 байт. Компьютер использует новый стек для новой информации чтобы данные сохранялись нормально и без помех (рис. 20).

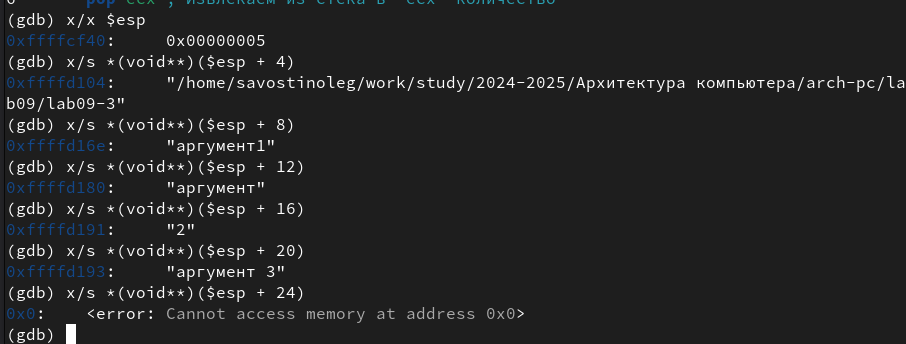


Рис. 20: Позиции стека

## 4.3 Выполнение самостоятельной работы.

1. Копирую файл из лабораторной работы 8 и переделываю текст файла, чтобы он работал с помощью подпрограммы (рис. 21).

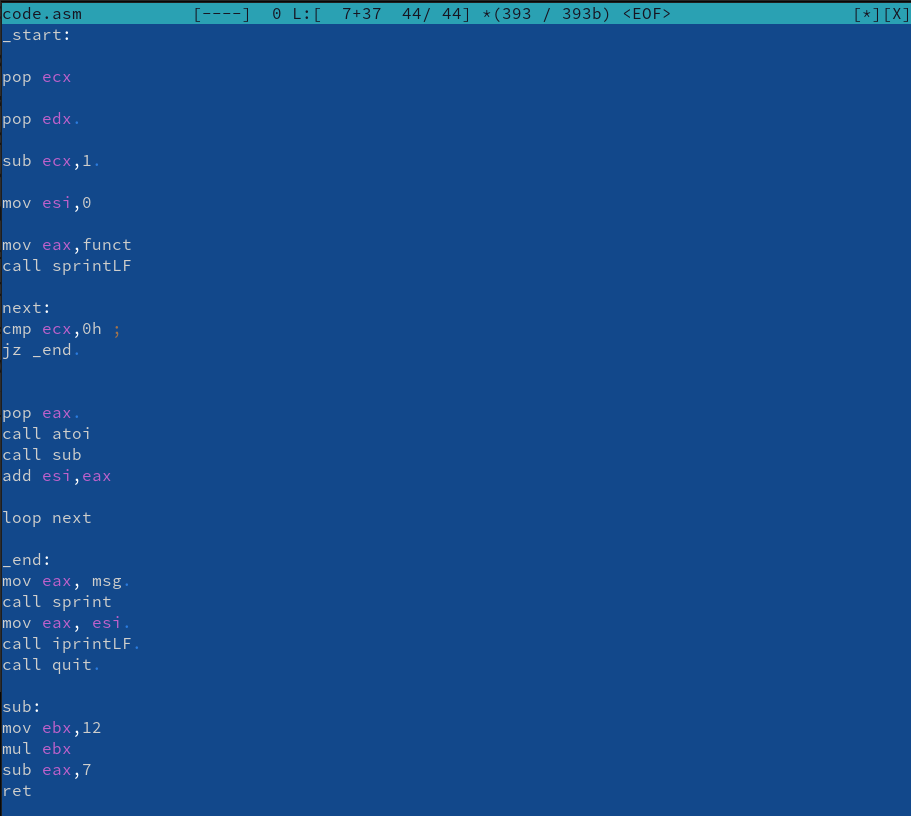


Рис. 21: Измененный код

Проверяю на правильность работы (рис. 22).

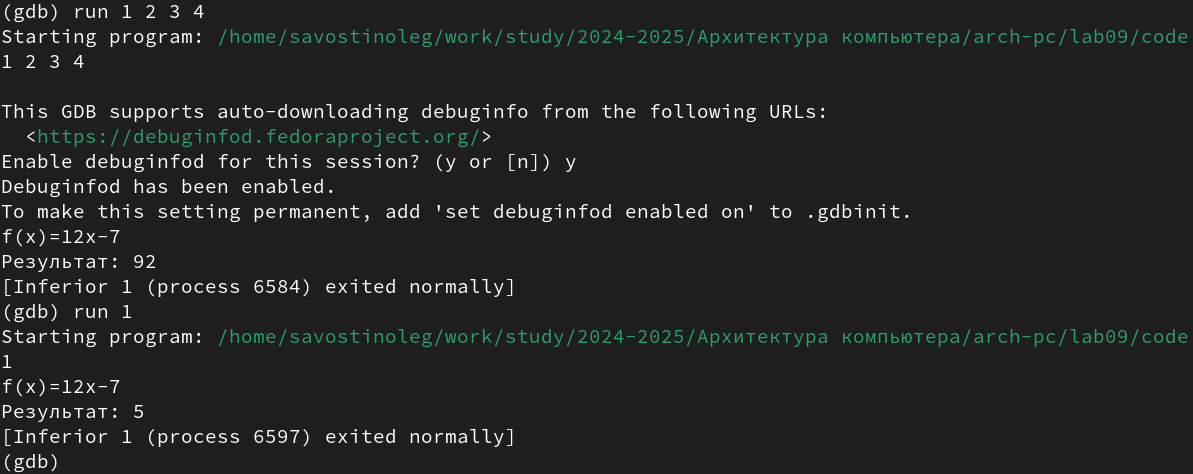


Рис. 22: Проверка на правильность

1. Создаю файл, и вставляю в него код из лабораторной работы. Проверяю на правильность. Выходит ошибка, так как (3+2)\*4+5 не равно 10. (рис. 23).

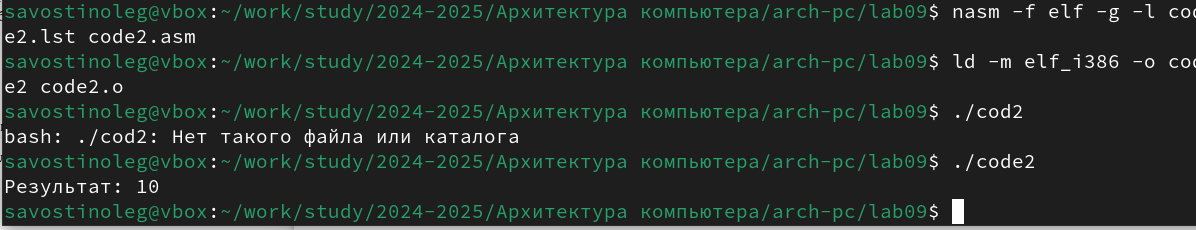


Рис. 23: Проверка на правильность файла из Листинга 9.3

Запускаю gdb, ошибка арифметическая (рис. 24). Анализирую код(рис. 25).

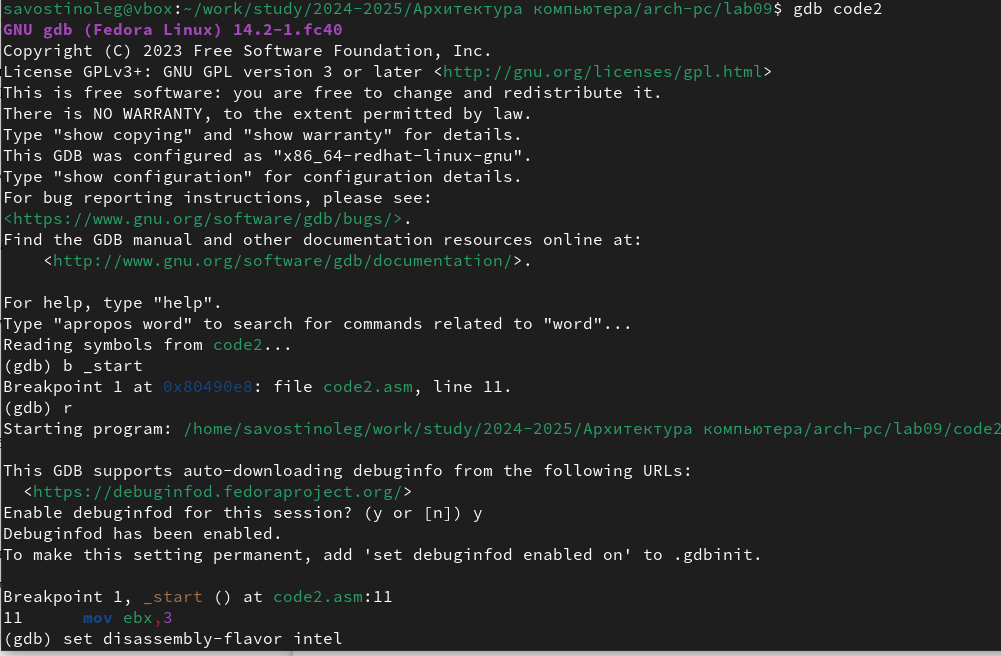


Рис. 24: Запуск gdb

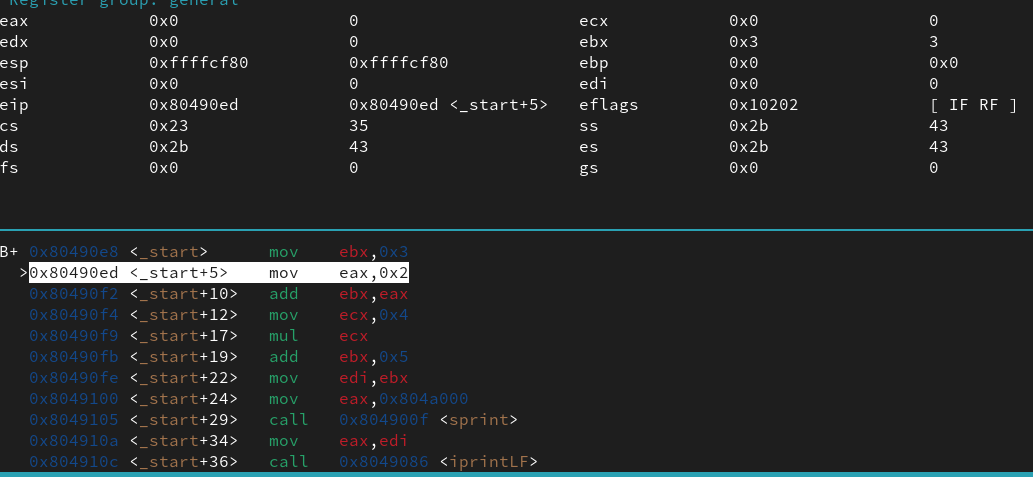


Рис. 25: Регистры.

Изменяю текст кода (рис. 26). Исправленный код(рис. 27).

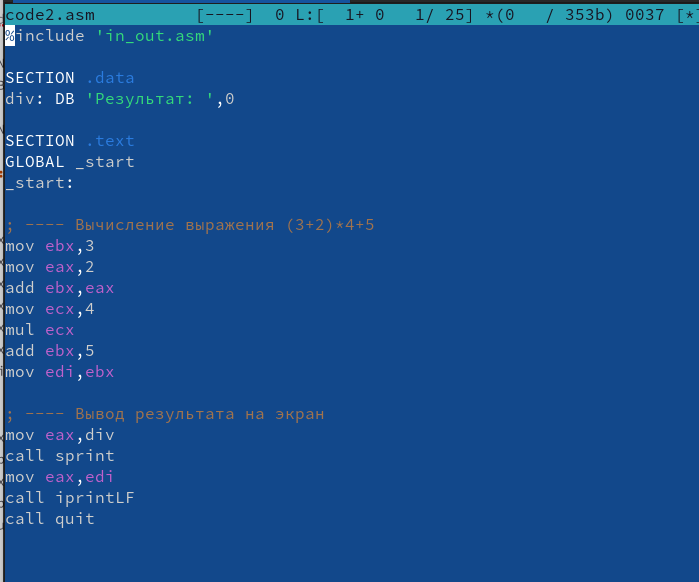


Рис. 26: Ошибочный код

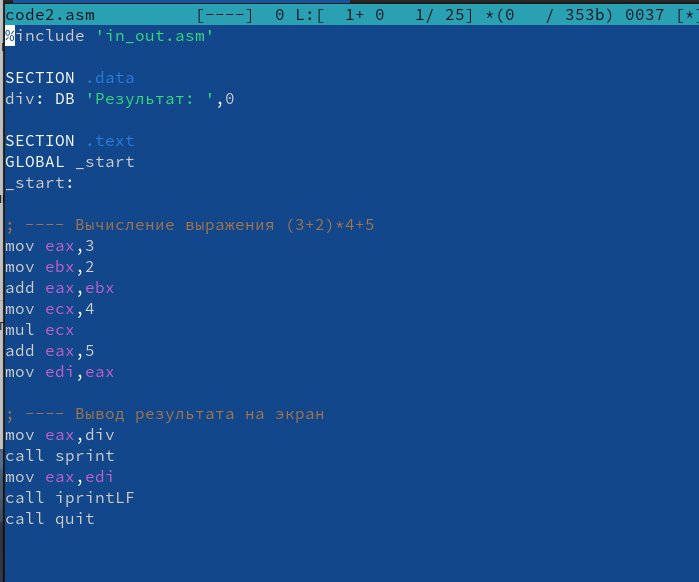


Рис. 27: Исправленный код

Проверяю на правильность. Все верно(рис. 28).

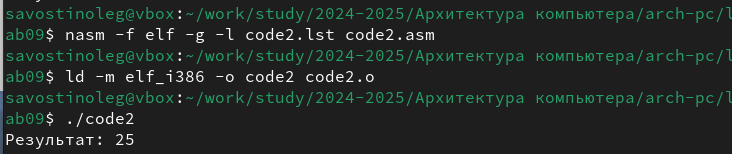


Рис. 28: Результат исправления кода

# 5 Выводы

В ходе работы, я приобрел навыки написания программ с использованием подпрограмм. Ознакомился с методами отладки при помощи GDB и его основными возможностями

# Список литературы

[Лабораторная работа №9](https://esystem.rudn.ru/pluginfile.php/2089551/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%969.%20%D0%9F%D0%BE%D0%BD%D1%8F%D1%82%D0%B8%D0%B5%20%D0%BF%D0%BE%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B.%20%D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D1%87%D0%B8%D0%BA%20..pdf)