Отчёт по лабораторной работе №9

дисциплина: Архитектура компьютера

Баранова Анна Андреевна

Содержание

# 1 Цель работы

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

# 2 Задание

В ходе выполнения данной лабораторной работы необходимо изучить:

* Понятие об отладке;
* методы отладки;
* основные возможности отладчика GDB;
* понятие подпрограммы.

Выполнив эту работу, мы приобретём навыки написания программ с использованием подпрограмм. Познакомимся с методами отладки при помощи GDB и его основными возможностями.

# 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.

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 изменять 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:

* Breakpoint — точка останова (остановка происходит, когда выполнение доходит до определённой строки, адреса или процедуры, отмеченной программистом);
* Watchpoint — точка просмотра (выполнение программы приостанавливается, если программа обратилась к определённой переменной: либо считала её значение, либо изменила его).

Точки останова устанавливаются в отладчике на время сеанса работы с кодом программы, т.е. они сохраняются до выхода из программы-отладчика или до смены отлаживаемой программы.

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. Отладчик не содержит собственного графического пользовательского интерфейса и использует стандартный текстовый интерфейс консоли. Однако для GDB существует несколько сторонних графических надстроек, а кроме того, некоторые интегрированные среды разработки используют его в качестве базовой подсистемы отладки.

Отладчик GDB (как и любой другой отладчик) позволяет увидеть, что происходит «внутри» программы в момент её выполнения или что делает программа в момент сбоя.

GDB может выполнять следующие действия:

* начать выполнение программы, задав всё, что может повлиять на её поведение;
* остановить программу при указанных условиях;
* исследовать, что случилось, когда программа остановилась;
* изменить программу так, чтобы можно было поэкспериментировать с устранением эффектов одной ошибки и продолжить выявление других.

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.

Если в программе встречается одинаковый участок кода, его можно оформить в виде подпрограммы, а во всех нужных местах поставить её вызов. При этом подпрограмма будет содержаться в коде в одном экземпляре, что позволит уменьшить размер кода всей программы.

# 4 Выполнение лабораторной работы

## 4.1 Реализация подпрограмм в NASM

Создадим каталог для выполнения лабораторной работы № 9, перейдём в него и создадим файл lab09-1.asm и также создадим копию файла in\_out.asm (рис. 1), (рис. 2), (рис. 3).

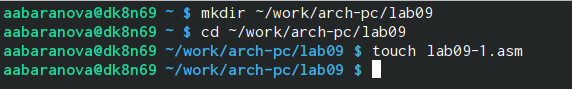


Рис. 1: Создание каталога для программ лабораторной работы № 8 и создание в нём файла lab8-1.asm

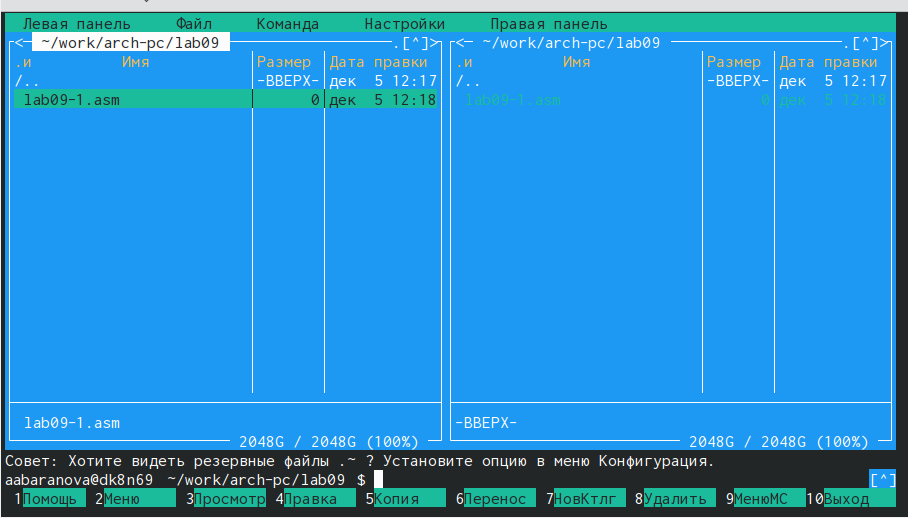


Рис. 2: Создание каталога для программ лабораторной работы № 8 и создание в нём файла lab8-1.asm

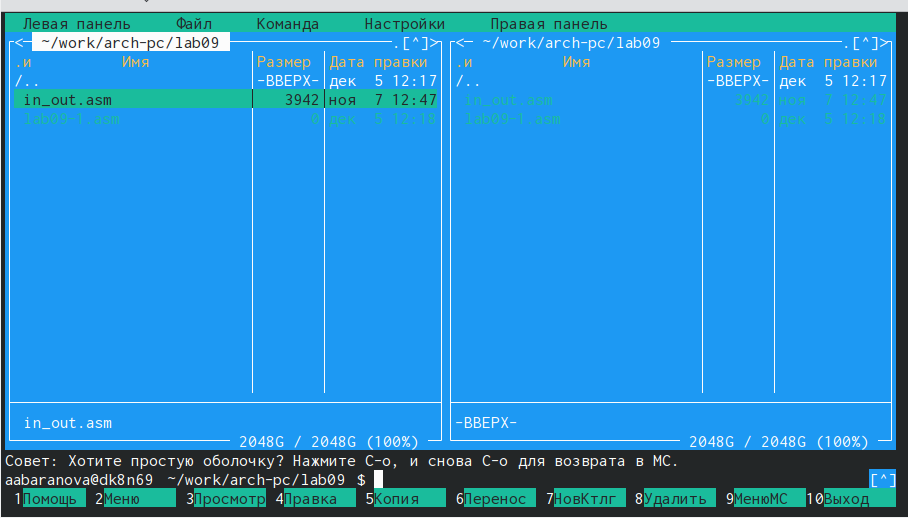


Рис. 3: Копирование файла in\_out.asm в каталог с файлом lab8-1.asm с помощью функциональной клавиши F5

Введём в файл lab09-1.asm текст программы (рис. 4).

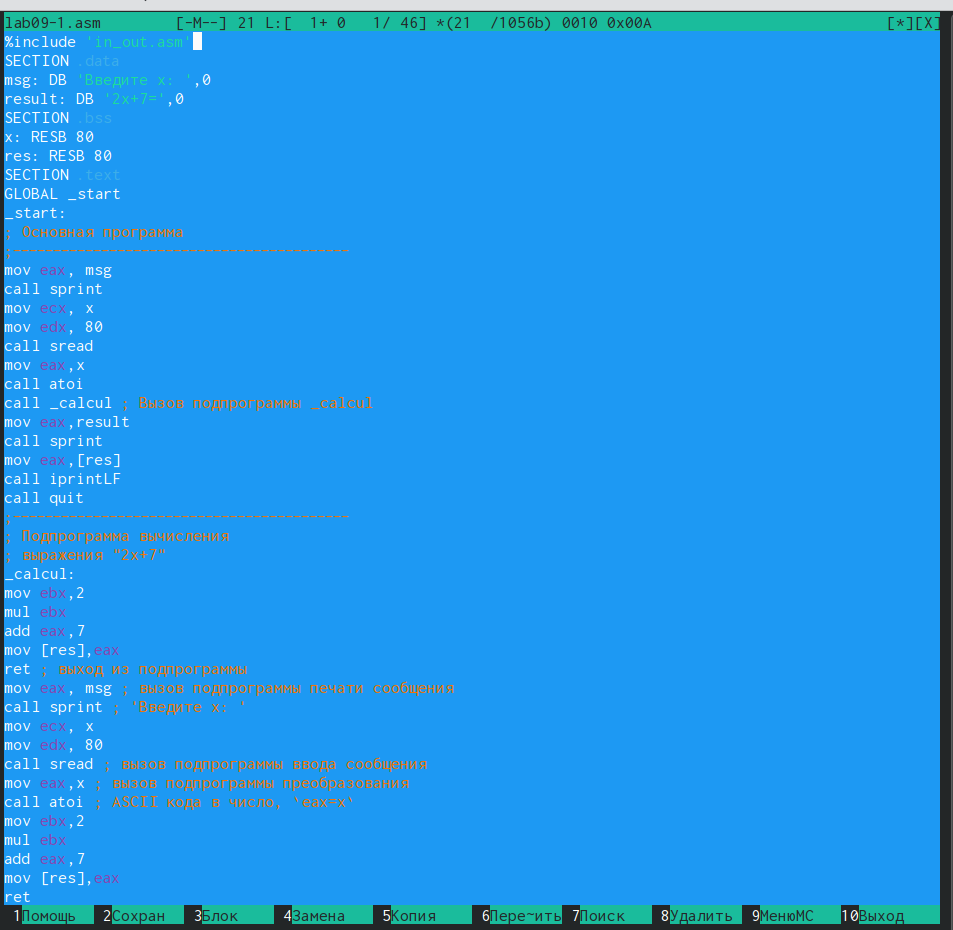


Рис. 4: Изменения в файле lab09-1.asm

Создадим исполняемый файл и запустим его (рис. 5), (рис. 6).

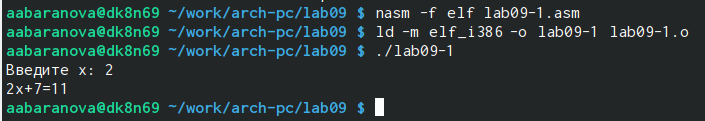


Рис. 5: Создание исполняемого файла и его запуск

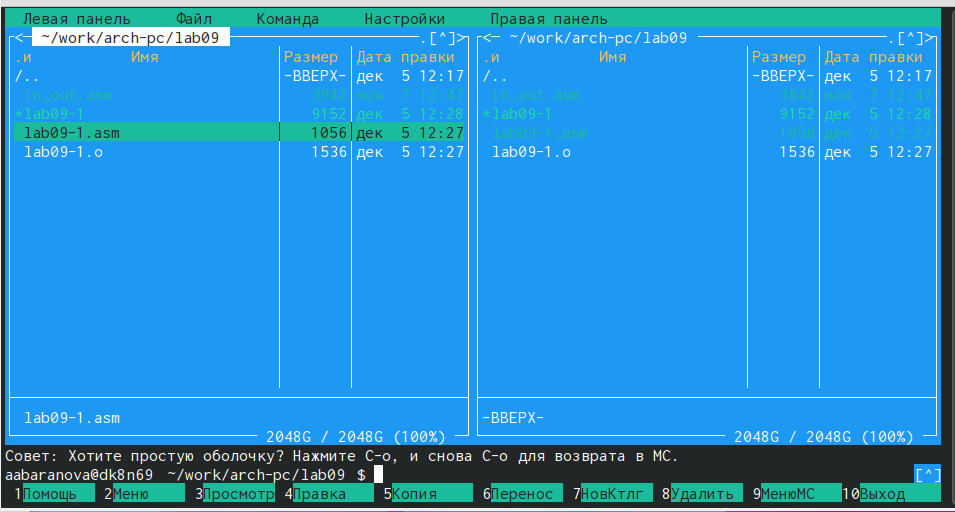


Рис. 6: Создание исполняемого файла и его запуск

Изменим текст программы файла lab09-1.asm (рис. 7).



Рис. 7: Изменения в файле lab09-1.asm

Создадим исполняемый файл и запустим его (рис. 8).

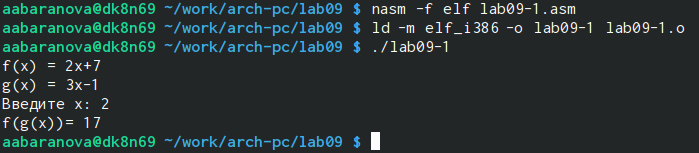


Рис. 8: Создание исполняемого файла и его запуск

## 4.2 Отладка программам с помощью GDB

Создадим файл lab09-2.asm в каталоге ~/work/arch-pc/lab09 (рис. 9), (рис. 10).

Рис. 9: Создание файла lab09-2.asm

Рис. 9: Создание файла lab09-2.asm

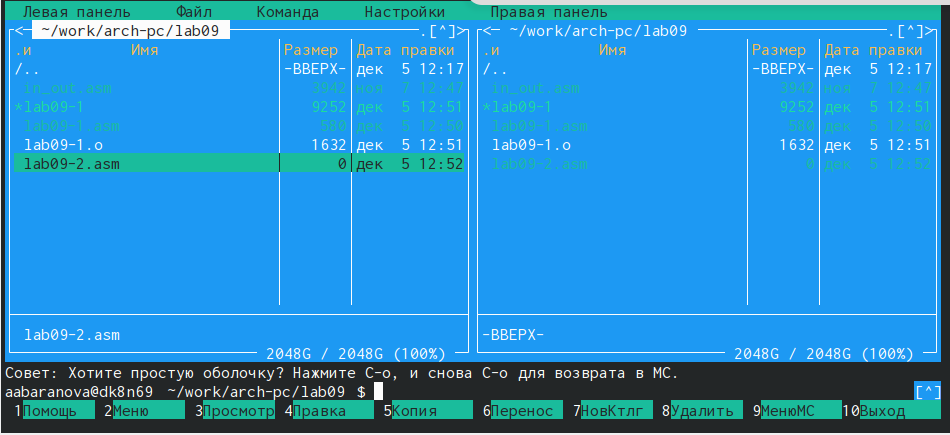


Рис. 10: Создание файла lab09-2.asm

Введём в файл lab09-2.asm текст программы (рис. 11).



Рис. 11: Изменения в файле lab09-2.asm

Получим исполняемый файл (рис. 12).

Рис. 12: Получение исполняемого файла

Рис. 12: Получение исполняемого файла

Загрузим исполняемый файл в отладчик gdb, проверим работу программы, запустив её в оболочке GDB с помощью команды run (рис. 13).

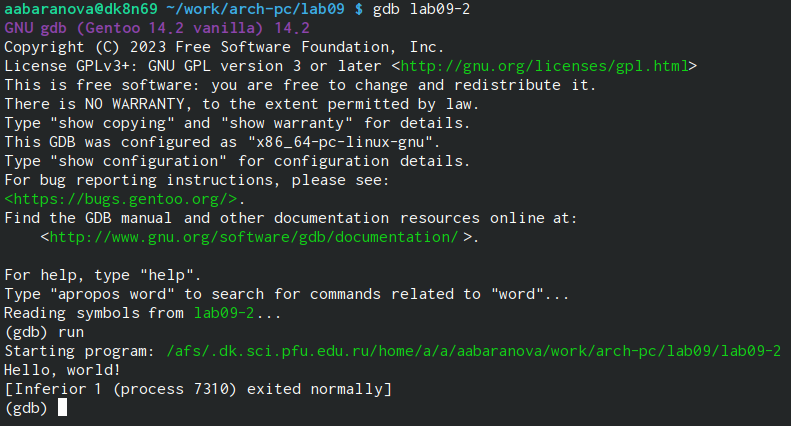


Рис. 13: Загрузка исполняемого файла в отладчик gdb и проверка работы программы с помощью команды run

Установим брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запустим её (рис. 14).

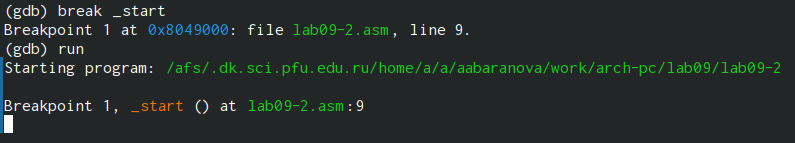


Рис. 14: Установка брейкпоинта на метку \_start, с которой начинается выполнение любой ассемблерной программы, и её запуск

Посмотрим дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. 15).



Рис. 15: Просмотр дисассимилированного кода программы с помощью команды disassemble начиная с метки \_start

Переключимся на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel. Отличие отображения синтаксиса машинных команд в режимах ATT и Intel заключается в командах, в диссамилированном отображении в командах используют % и $, а в Intel отображение эти символы не используются (рис. 16).



Рис. 16: Переключение на отображение команд с Intel’овским синтаксисом

Включим режим псевдографики для более удобного анализа программы (рис. 17).

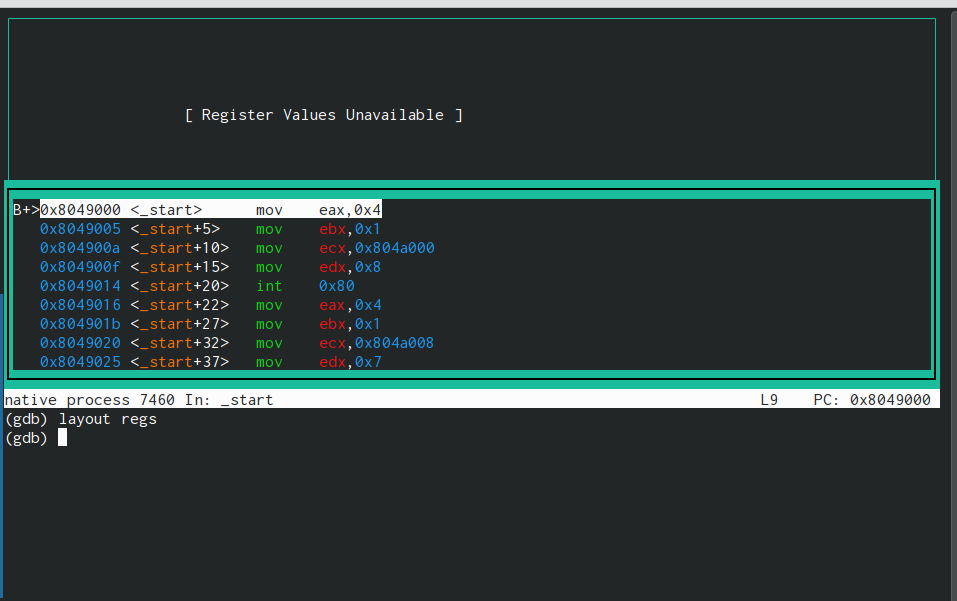


Рис. 17: Включение режима псевдографики

На предыдущих шагах была установлена точка останова по имени метки (\_start). Проверим это с помощью команды info breakpoints (кратко i b) (рис. 18).

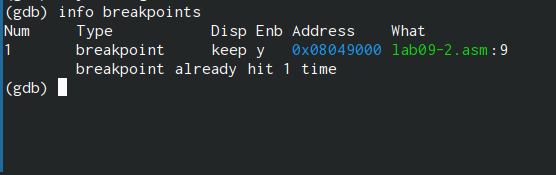


Рис. 18: Проверка установки точки останова

Определим адрес предпоследней инструкции (mov ebx,0x0) и установим точку останова (рис. 19).

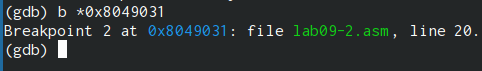


Рис. 19: Установка точки останова

Посмотрим информацию о всех установленных точках останова (рис. 20).

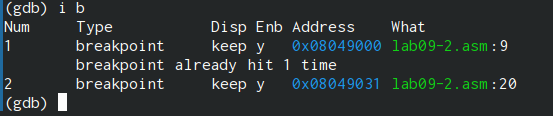


Рис. 20: Просмотр информации о всех установленных точках останова

Выполним 5 инструкций с помощью команды stepi (или si) и проследим за изменением значений регистров (рис. 21), (рис. 22), (рис. 23), (рис. 24), (рис. 25).

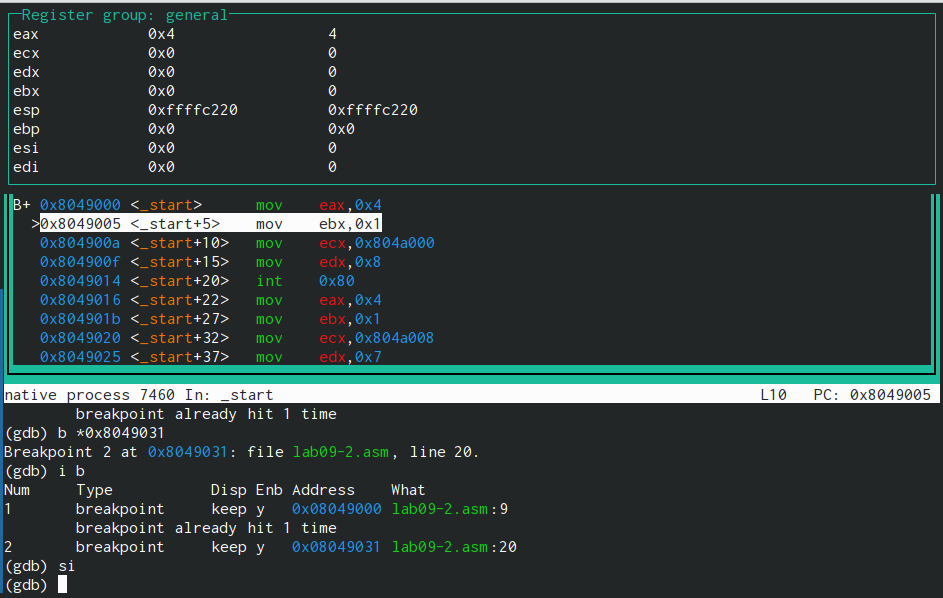


Рис. 21: Выполние инструкции с помощью команды stepi

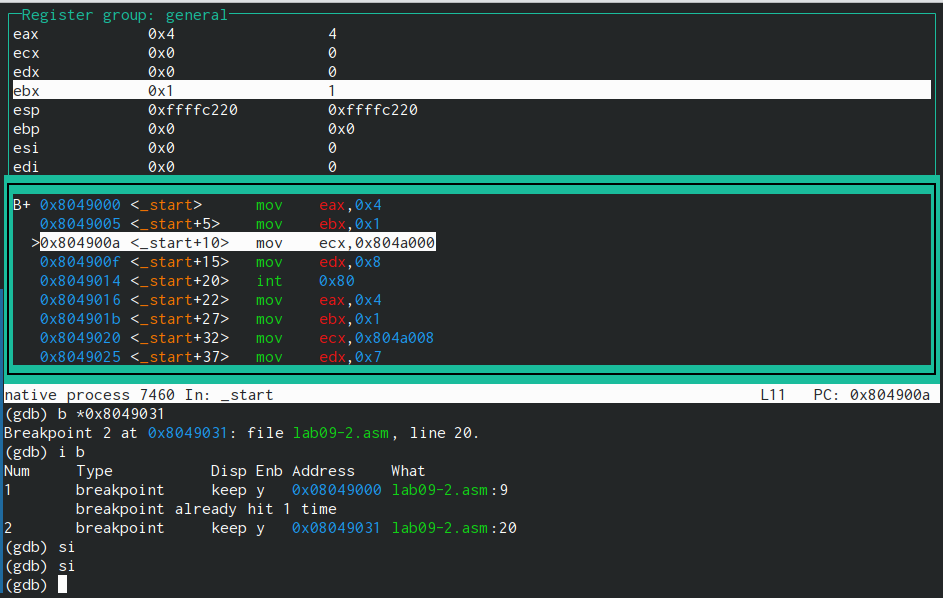


Рис. 22: Выполние инструкции с помощью команды stepi

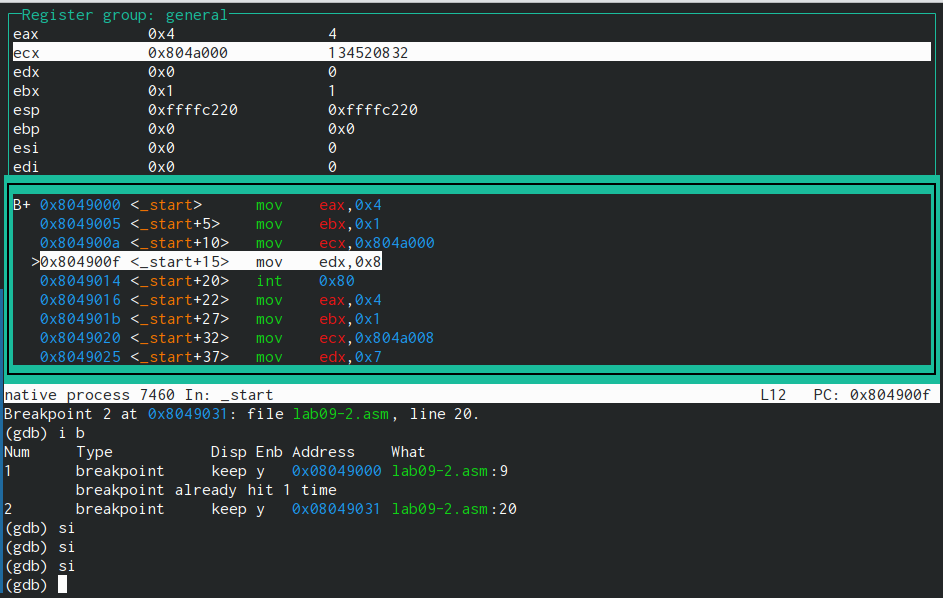


Рис. 23: Выполние инструкции с помощью команды stepi

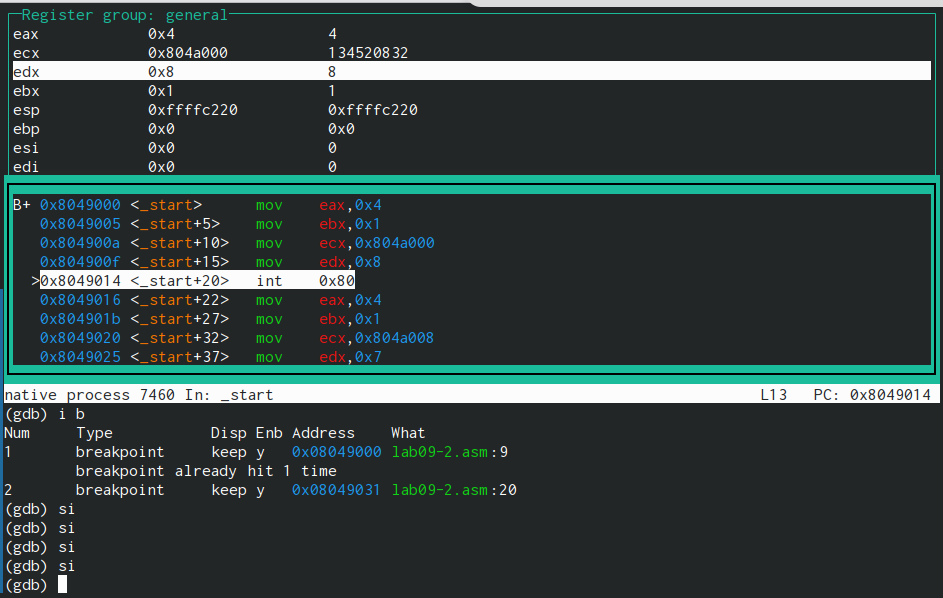


Рис. 24: Выполние инструкции с помощью команды stepi

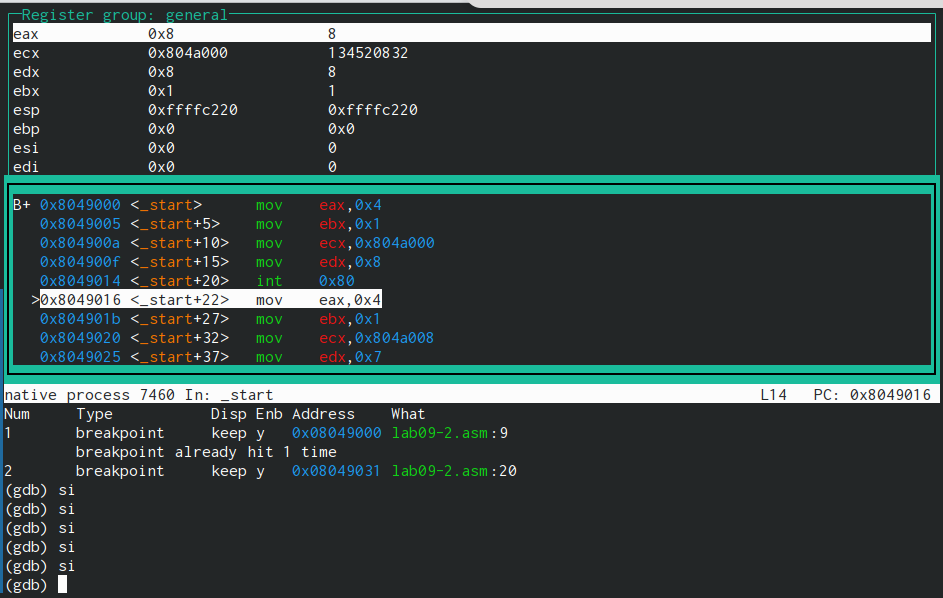


Рис. 25: Выполние инструкции с помощью команды stepi

Посмотрим содержимое регистров с помощью команды info registers (или i r) (рис. 26).

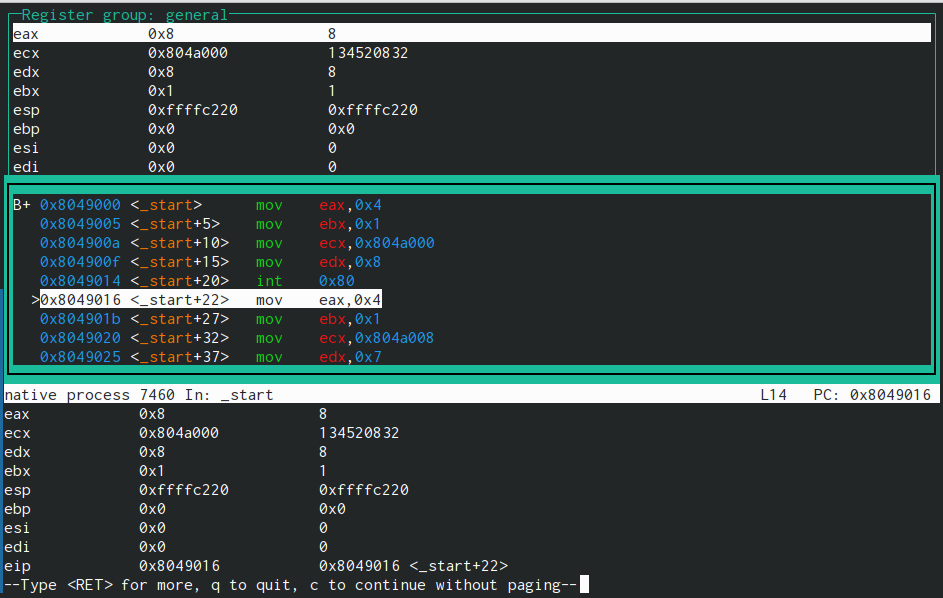


Рис. 26: Просмотр содержимого регистров с помощью команды info registers

Посмотрим значение переменной msg1 по имени (рис. 27).

Рис. 27: Просмотр значения переменной msg1 по имени

Рис. 27: Просмотр значения переменной msg1 по имени

Посмотрим значение переменной msg2 по адресу (рис. 28).

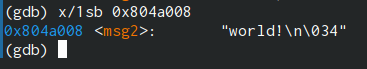


Рис. 28: Просмотр значения переменной msg2 по адресу

Изменим первый символ переменной msg1 (рис. 29).

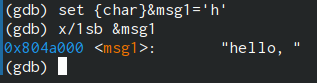


Рис. 29: Изменение первого символа переменной msg1

Изменим первый символ переменной msg2 (рис. 30).

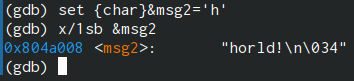


Рис. 30: Изменение первого символа переменной msg2

Выведем в шестнадцатеричном формате, в двоичном формате и в символьном виде значение регистра edx (рис. 31).

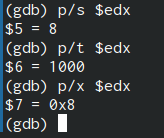


Рис. 31: Вывод в различных форматах значения регистра edx

С помощью команды set изменим значение регистра ebx. Команда выводит два разных значения так как в первый раз вносим значение 2, а во второй раз регистр равен двум, поэтому и значения разные (рис. 32).

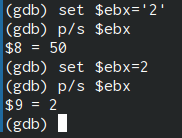


Рис. 32: Изменение значения регистра ebx

Завершим выполнение программы с помощью команды stepi (сокращенно si) и выйдем из GDB с помощью команды quit (сокращенно q) (рис. 33).

Рис. 33: Завершение выполнения программы и выход из GDB

Рис. 33: Завершение выполнения программы и выход из GDB

Скопируем файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки в файл с именем lab09-3.asm и создадим исполняемый файл(рис. 34).

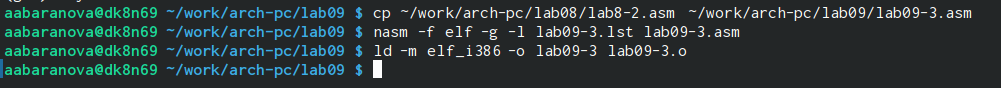


Рис. 34: Копирование файла lab8-2.asm в файл с именем lab09-3.asm и создание исполняемого файла

Загрузим исполняемый файл в отладчик, указав аргументы (рис. 35).

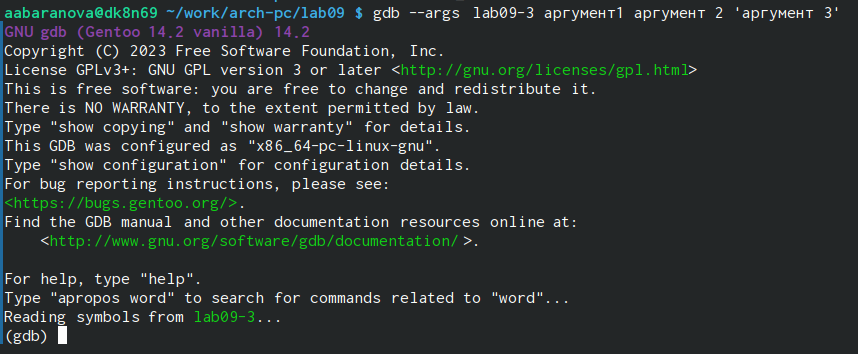


Рис. 35: Загрузка исполняемого файла в отладчик

Установим точку останова перед первой инструкцией в программе и запустим её (рис. 36).

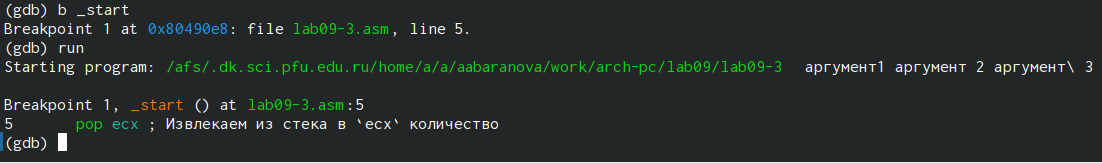


Рис. 36: Установка точки останова перед первой инструкцией в программе и её запуск

Проверим адрес вершины стека и убедимся, что там хранится 5 элементов (рис. 37).

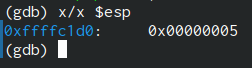


Рис. 37: Проверка адреса вершины стека

Посмотрим все позиции стека. По первому адрему хранится адрес, в остальных адресах хранятся элементы. Элементы расположены с интервалом в 4 единицы, так как стек может хранить до 4 байт, и для того чтобы данные сохранялись нормально и без помех, компьютер использует новый стек для новой информации (рис. 38).

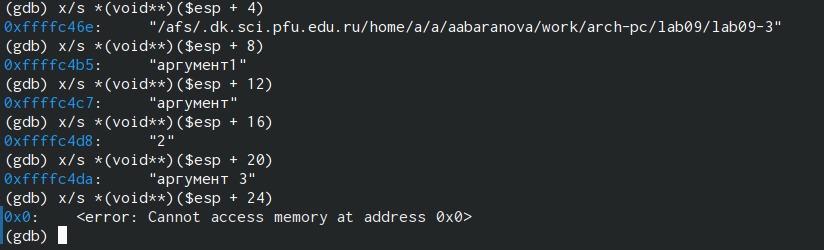


Рис. 38: Просмотр всех позиций стека

## 4.3 Задание для самостоятельной работы

Преобразуем программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции f(x) как подпрограмму (рис. 39), (рис. 40), (рис. 41), (рис. 42).

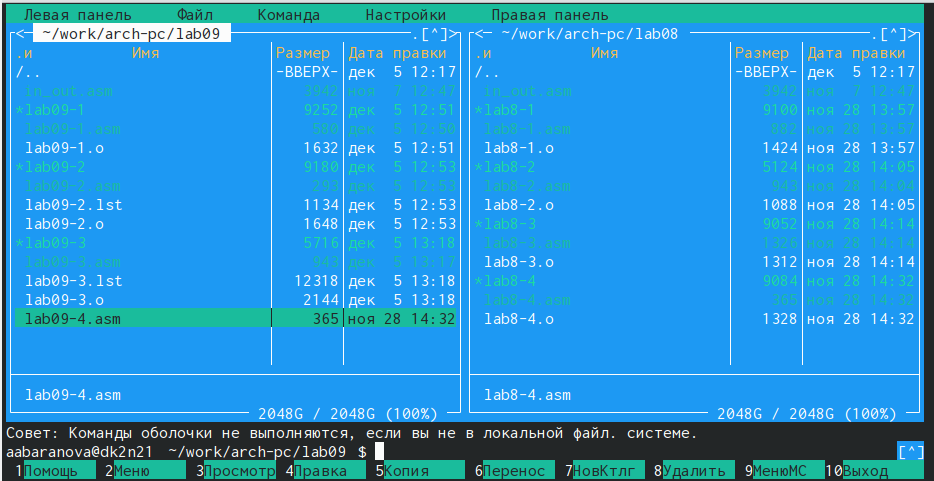


Рис. 39: Преобразование программы из лабораторной работы №8

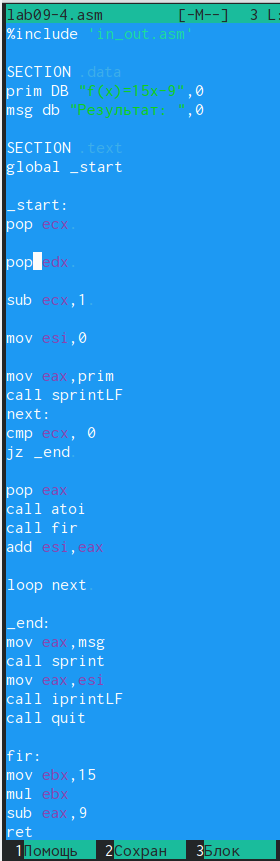


Рис. 40: Преобразование программы из лабораторной работы №8

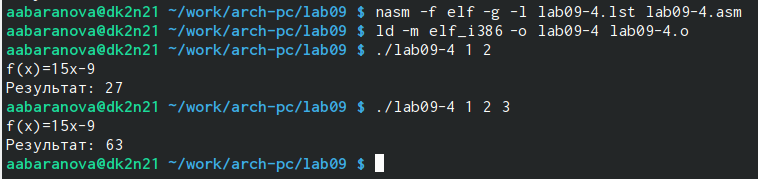


Рис. 41: Преобразование программы из лабораторной работы №8

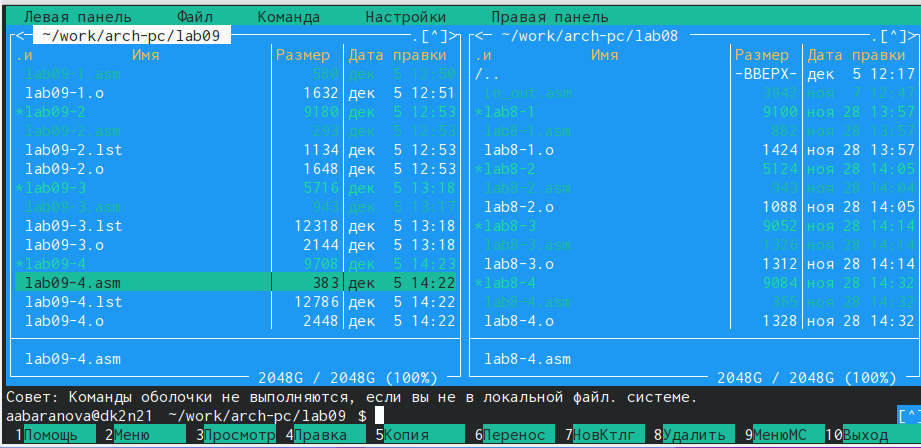


Рис. 42: Преобразование программы из лабораторной работы №8

В листинге 9.3 приведена программа вычисления выражения (3 + 2) ∗ 4 + 5. При запуске данная программа даёт неверный результат. Проверим это. С помощью отладчика GDB, анализируя изменения значений регистров, определим ошибку и исправим её (рис. 43), (рис. 44), (рис. 45), (рис. 46), рис. 47), рис. 48), рис. 49), рис. 50), рис. 51), рис. 52).

Рис. 43: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

Рис. 43: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

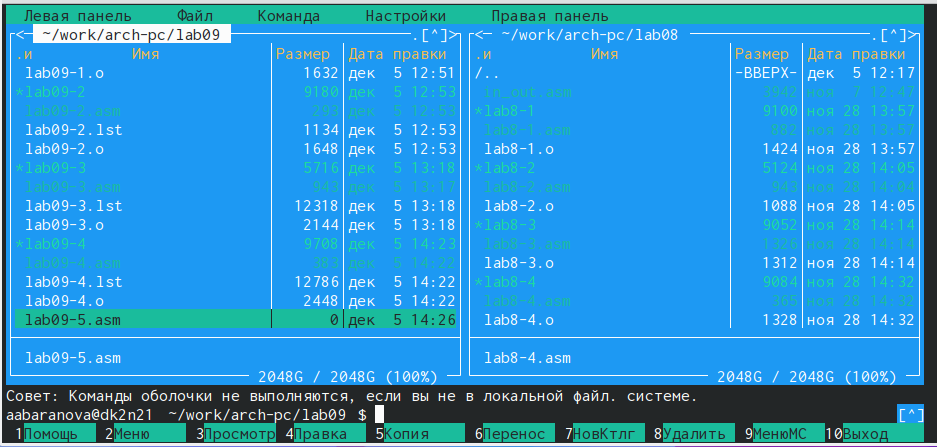


Рис. 44: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

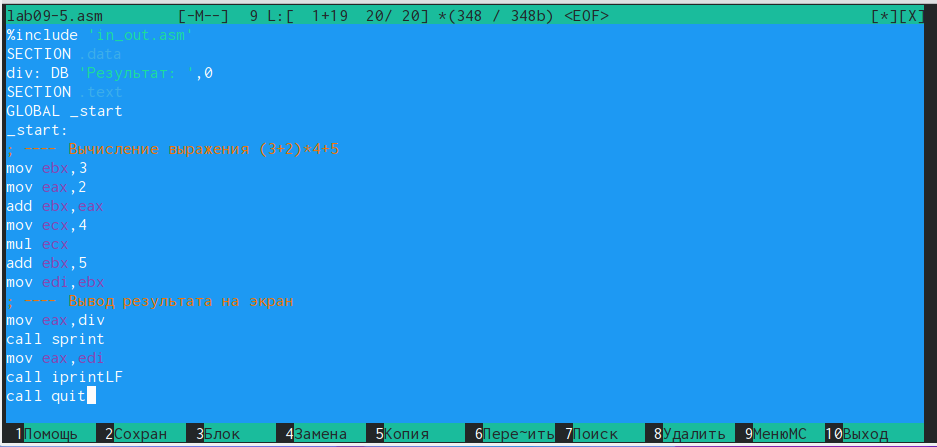


Рис. 45: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

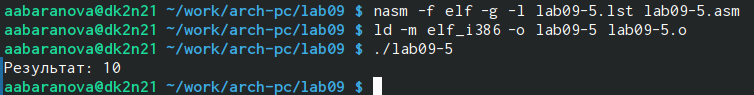


Рис. 46: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

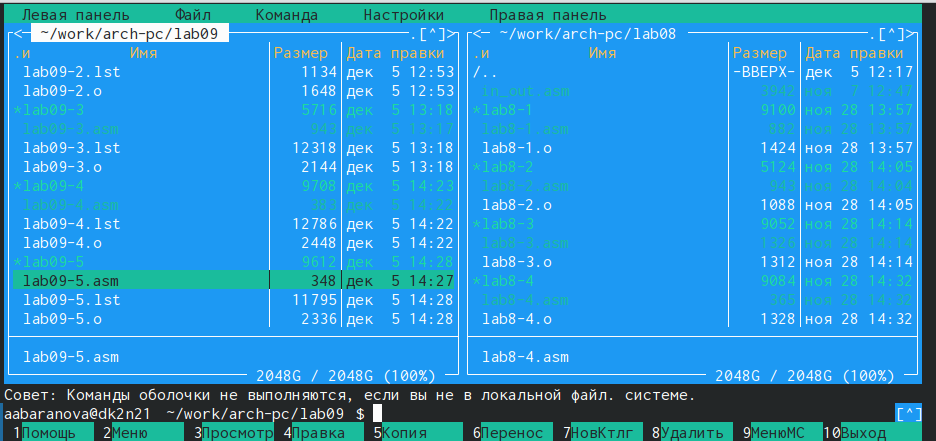


Рис. 47: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

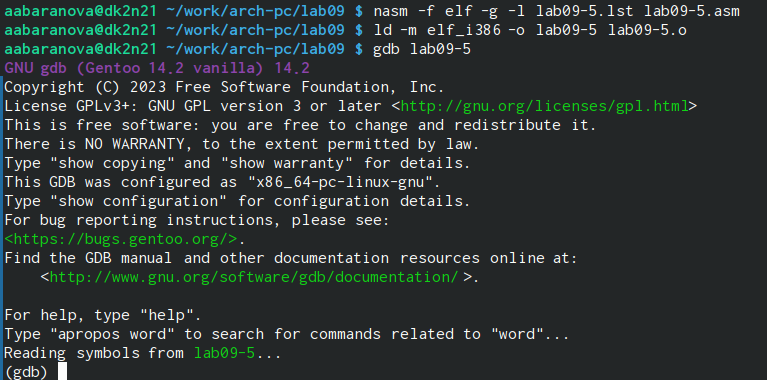


Рис. 48: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

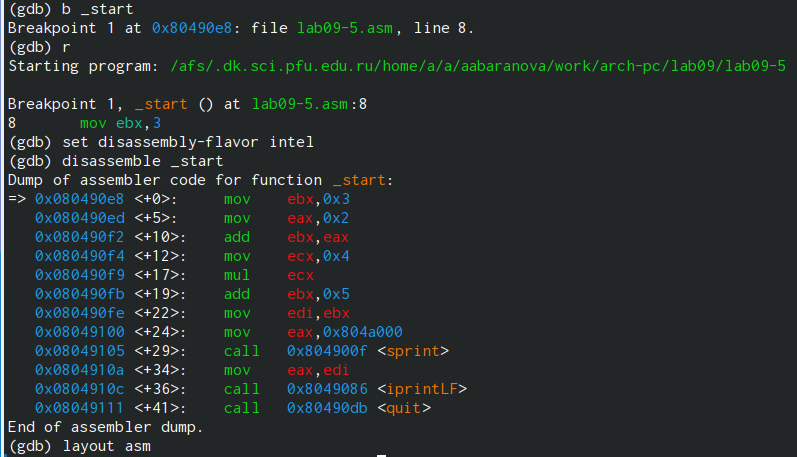


Рис. 49: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

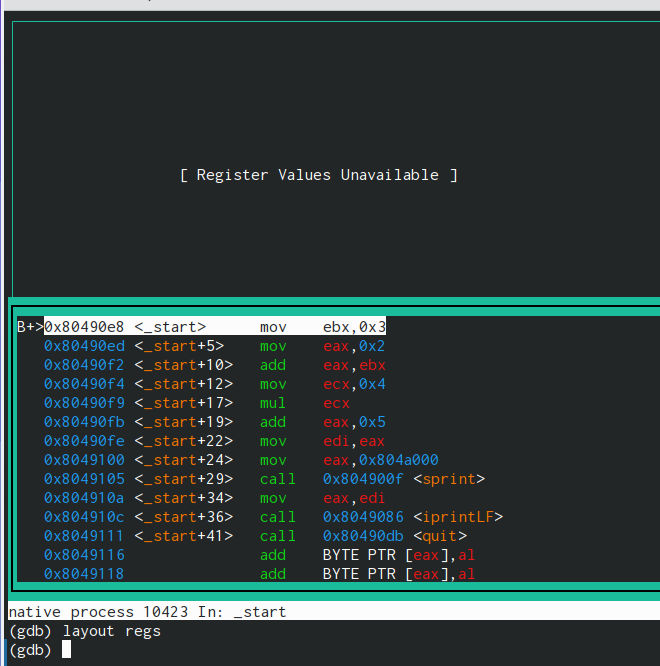


Рис. 50: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

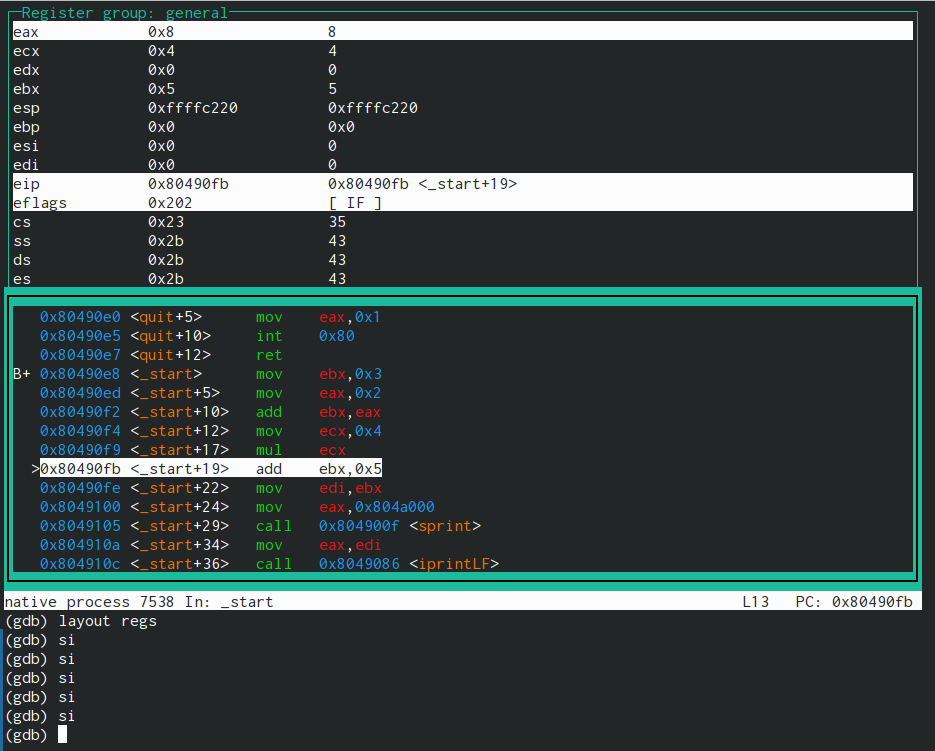


Рис. 51: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

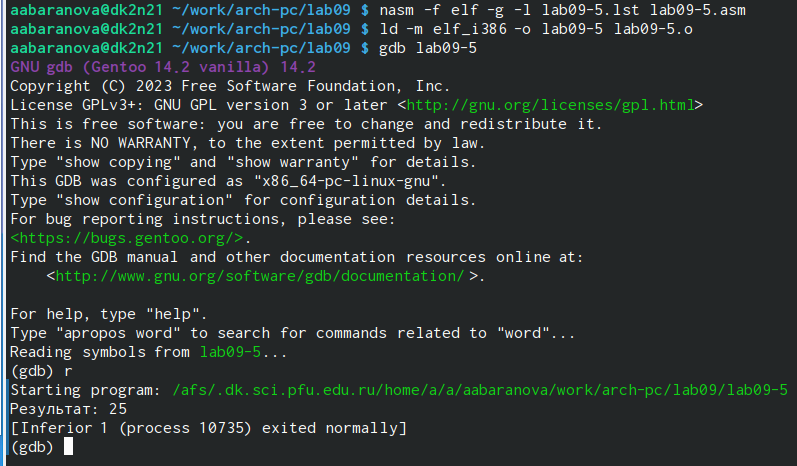


Рис. 52: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

# 5 Выводы

В ходе выполнения данной лабораторной работы были приобретены навыки написания программ с использованием подпрограмм. Также познакомились с методами отладки при помощи GDB и его основными возможностями.