Отчёт по лабораторной работе №9

дисциплина: Архитектура компьютера

Баранова Анна Андреевна

Содержание

List of Figures

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. 4.1), (рис. 4.2), (рис. 4.3).

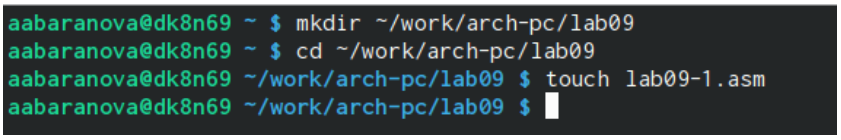

Рис. 4.1: Создание каталога для программ лабораторной работы № 8 и создание в нём файла lab8-1.asm

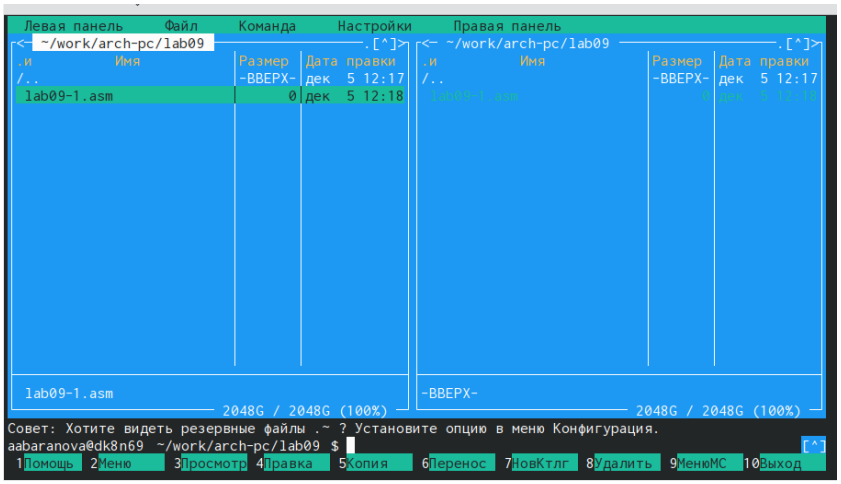

Рис. 4.2: Создание каталога для программ лабораторной работы № 8 и создание в нём файла lab8-1.asm

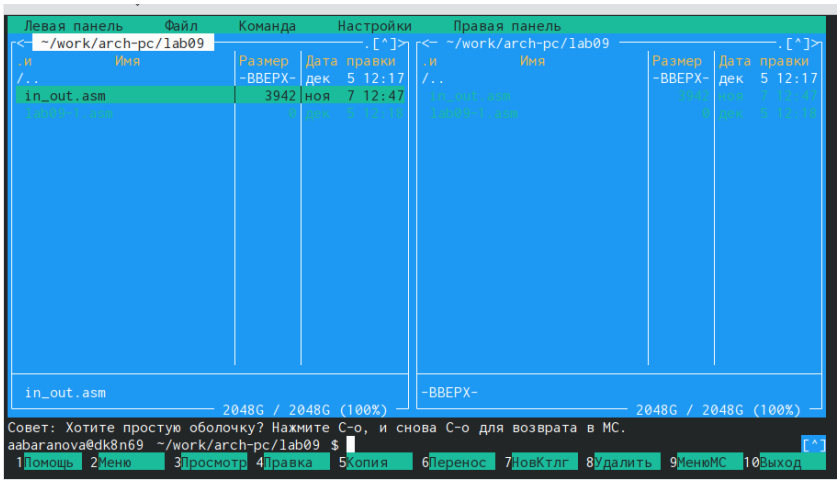

Рис. 4.3: Копирование файла in_out.asm в каталог с файлом lab8-1.asm с помощью функциональной клавиши F5

Введём в файл lab09-1.asm текст программы (рис. 4.4).

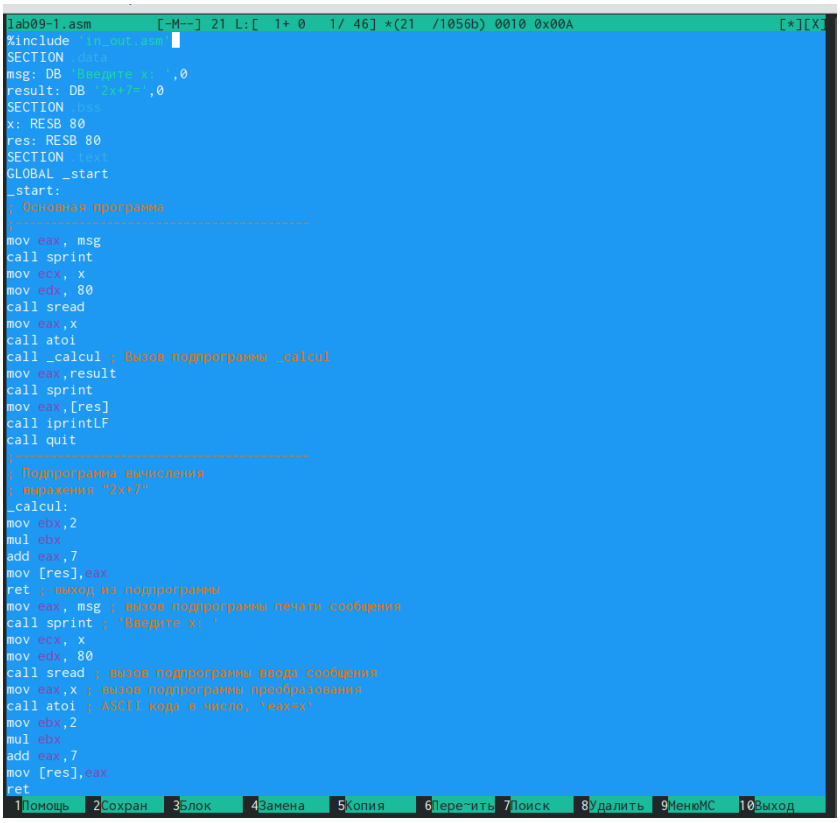

Рис. 4.4: Изменения в файле lab09-1.asm

Создадим исполняемый файл и запустим его (рис. 4.5), (рис. 4.6).

Рис. 4.5: Создание исполняемого файла и его запуск

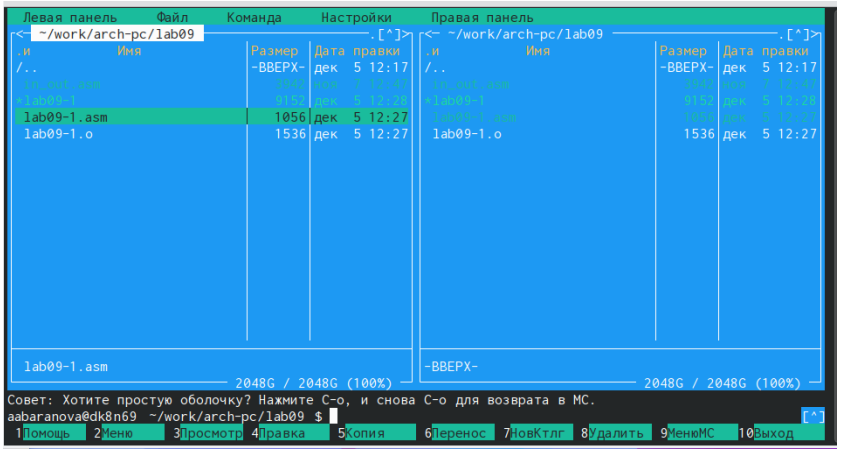

Рис. 4.6: Создание исполняемого файла и его запуск

Изменим текст программы файла lab09-1.asm (рис. 4.7).

Рис. 4.7: Изменения в файле lab09-1.asm

Создадим исполняемый файл и запустим его (рис. 4.8).

```
aabaranova@dk8n69 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
aabaranova@dk8n69 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
aabaranova@dk8n69 ~/work/arch-pc/lab09 $ ./lab09-1
f(x) = 2x+7
g(x) = 3x-1
Введите x: 2
f(g(x))= 17
aabaranova@dk8n69 ~/work/arch-pc/lab09 $
```

Рис. 4.8: 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. 4.9), (рис. 4.10).

```
aabaranova@dk8n69 ~/work/arch-pc/lab09 $ touch lab09-2.asm
aabaranova@dk8n69 ~/work/arch-pc/lab09 $
```

Рис. 4.9: Создание файла lab09-2.asm

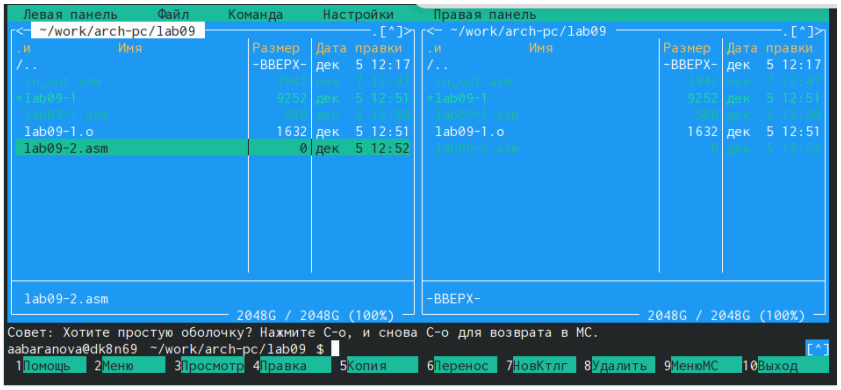

Рис. 4.10: Создание файла lab09-2.asm

Введём в файл lab09-2.asm текст программы (рис. 4.11).

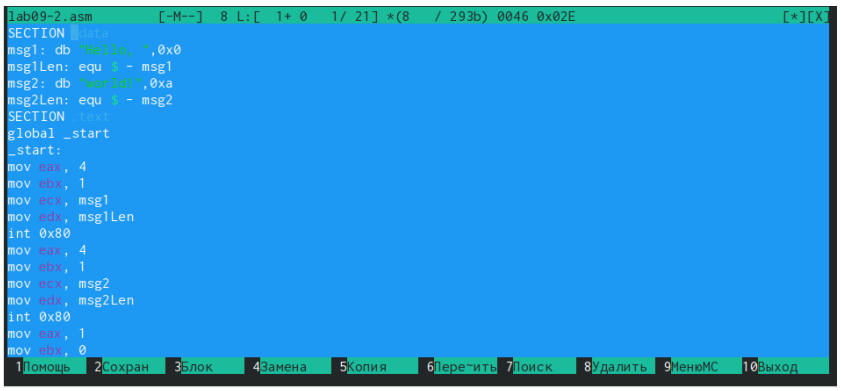

Рис. 4.11: Изменения в файле lab09-2.asm

Получим исполняемый файл (рис. 4.12).

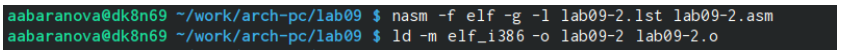

Рис. 4.12: Получение исполняемого файла

Загрузим исполняемый файл в отладчик gdb, проверим работу программы, запустив её в оболочке GDB с помощью команды run (рис. 4.13).

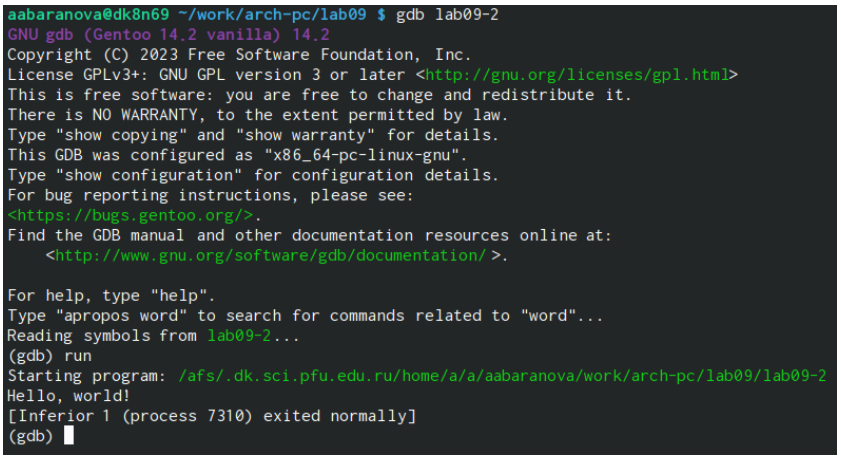

Рис. 4.13: Загрузка исполняемого файла в отладчик gdb и проверка работы программы с помощью команды run

Установим брейкпоинт на метку _start, с которой начинается выполнение любой ассемблерной программы, и запустим её (рис. 4.14).

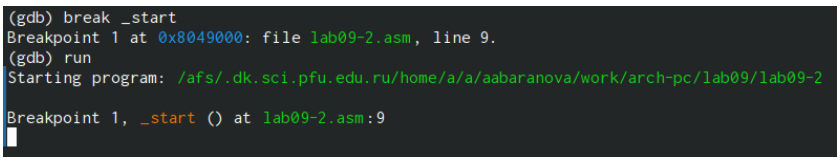

Рис. 4.14: Установка брейкпоинта на метку _start, с которой начинается выполнение любой ассемблерной программы, и её запуск

Посмотрим дисассимилированный код программы с помощью команды disassemble начиная с метки _start (рис. 4.15).

Рис. 4.15: 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. 4.16).

Рис. 4.16: Переключение на отображение команд с Intel'овским синтаксисом

Включим режим псевдографики для более удобного анализа программы (рис. 4.17).

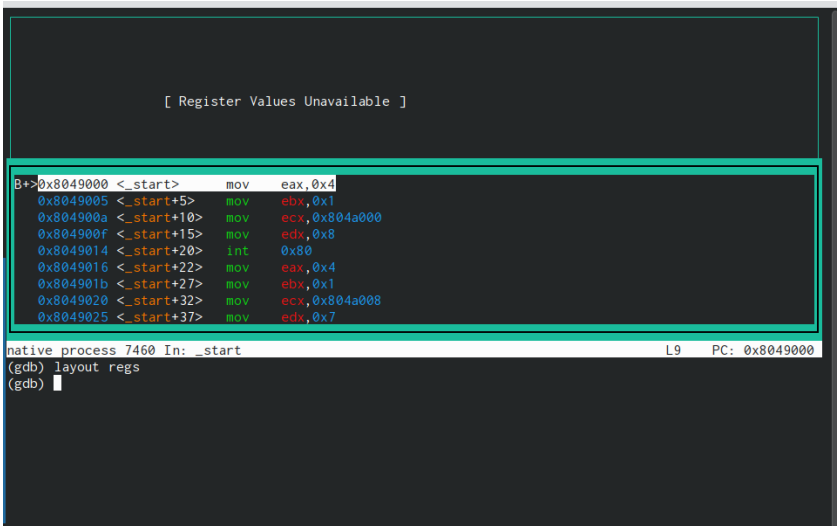

Рис. 4.17: Включение режима псевдографики

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

4.18).

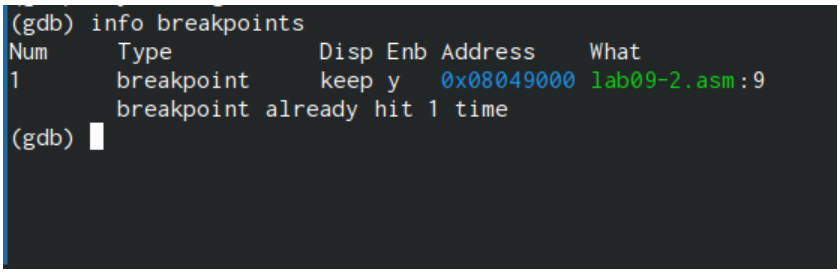

Рис. 4.18: Проверка установки точки останова

Определим адрес предпоследней инструкции (mov ebx,0x0) и установим точку останова (рис. 4.19).

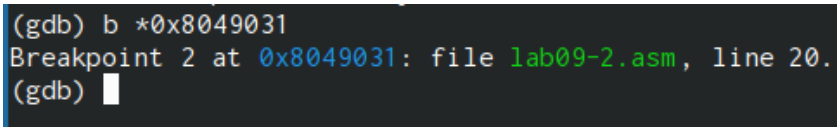

Рис. 4.19: Установка точки останова

Посмотрим информацию о всех установленных точках останова (рис. 4.20).

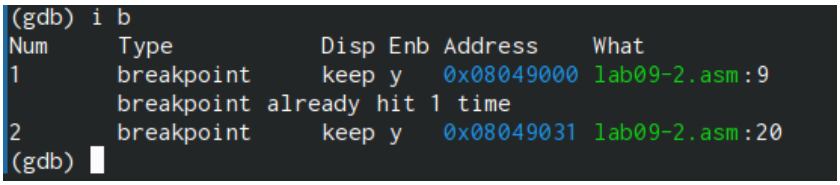

Рис. 4.20: Просмотр информации о всех установленных точках останова

Выполним 5 инструкций с помощью команды stepi (или si) и проследим за изменением значений регистров (рис. 4.21), (рис. 4.22), (рис. 4.23), (рис. 4.24), (рис. 4.25).

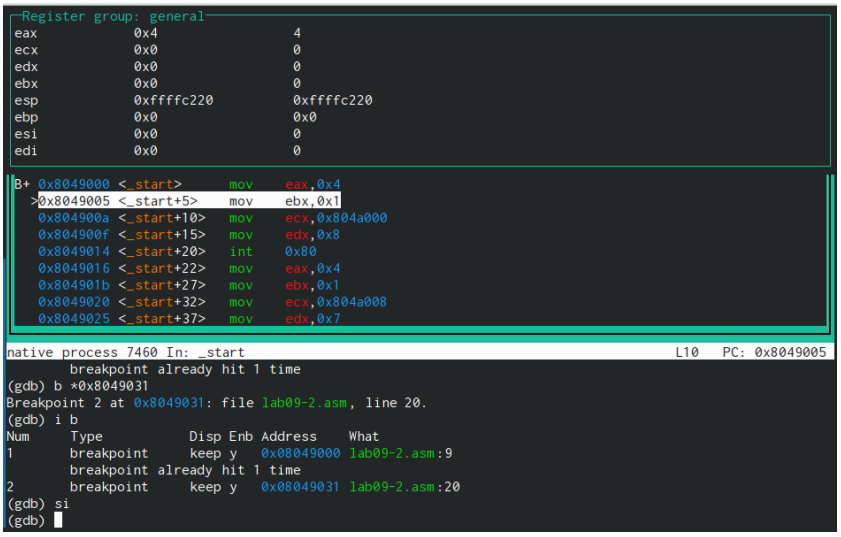

Рис. 4.21: Выполние инструкции с помощью команды stepi

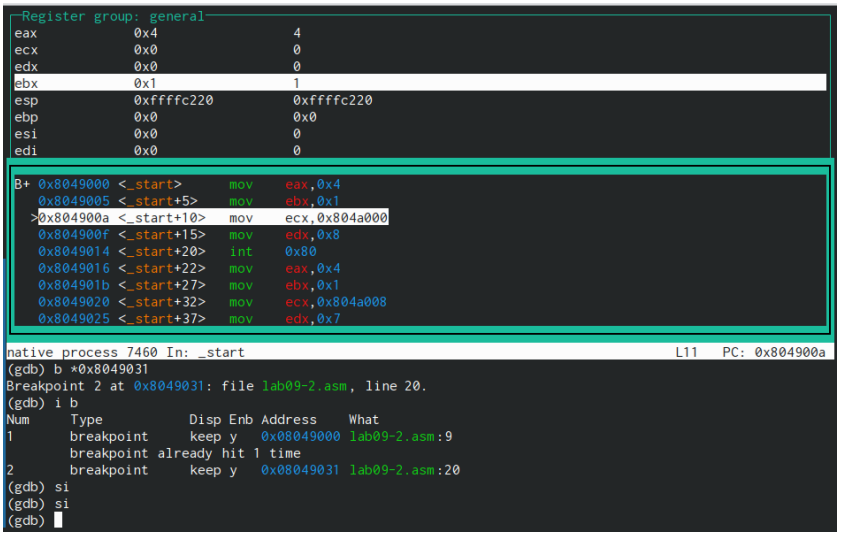

Рис. 4.22: Выполние инструкции с помощью команды stepi

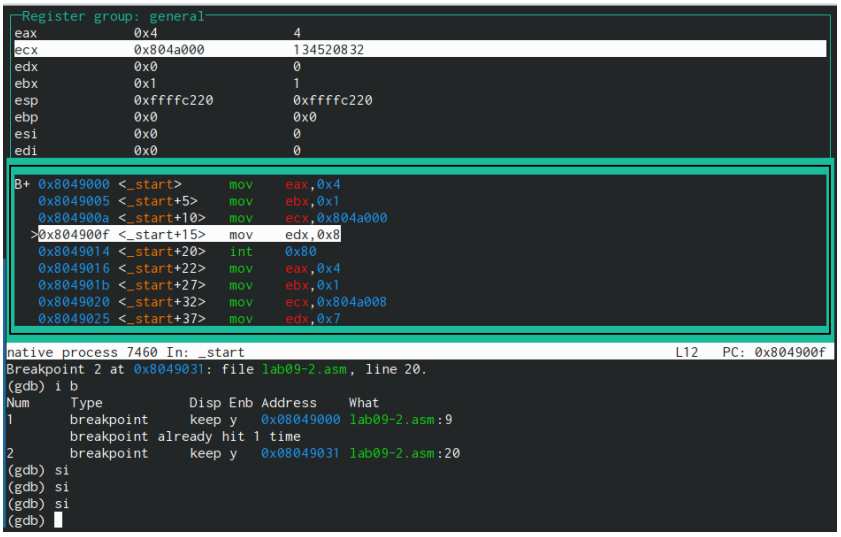

Рис. 4.23: Выполние инструкции с помощью команды stepi

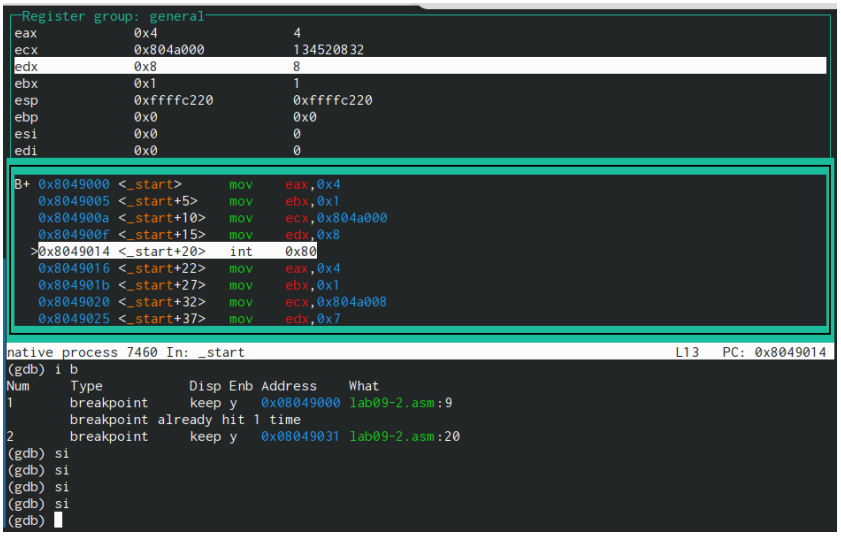

Рис. 4.24: Выполние инструкции с помощью команды stepi

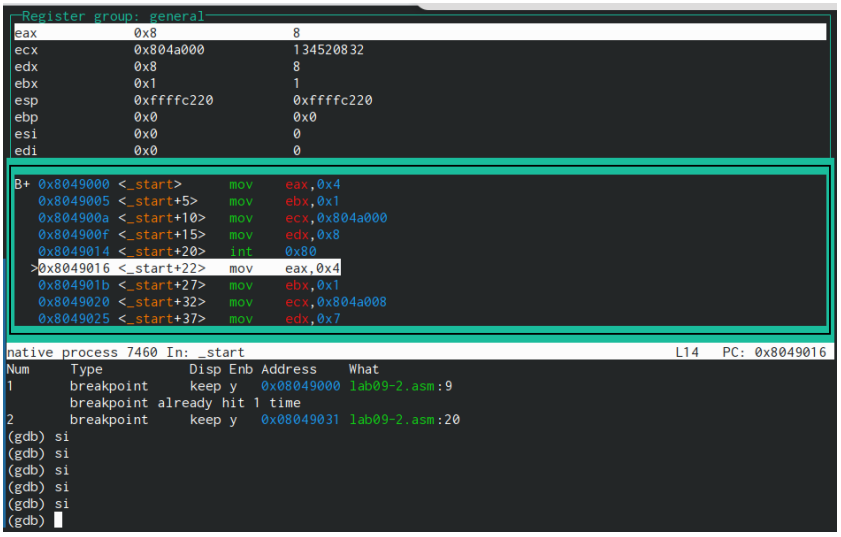

Рис. 4.25: Выполние инструкции с помощью команды stepi

Посмотрим содержимое регистров с помощью команды info registers (или i r) (рис. 4.26).

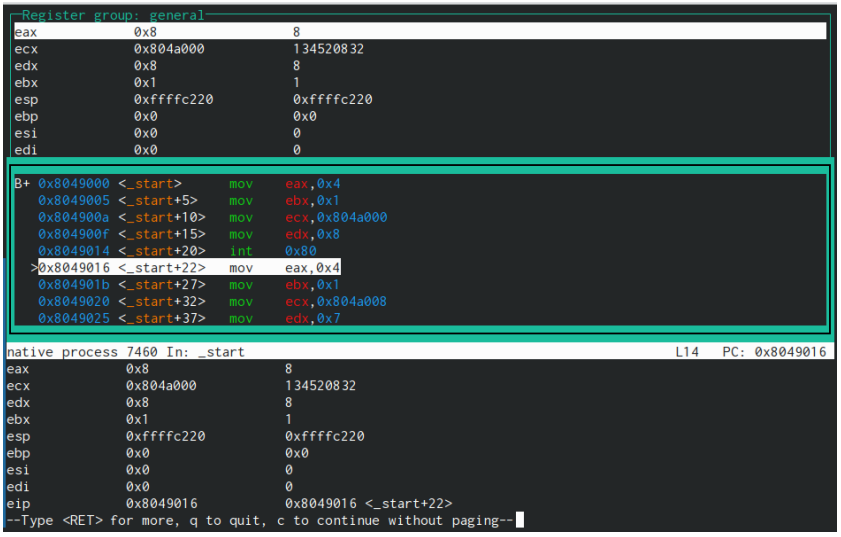

Рис. 4.26: Просмотр содержимого регистров с помощью команды info registers

Посмотрим значение переменной msg1 по имени (рис. 4.27).

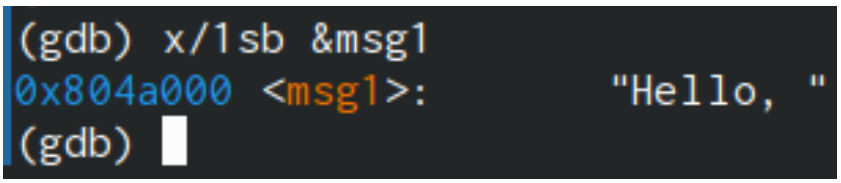

Рис. 4.27: Просмотр значения переменной msg1 по имени

Посмотрим значение переменной msg2 по адресу (рис. 4.28).

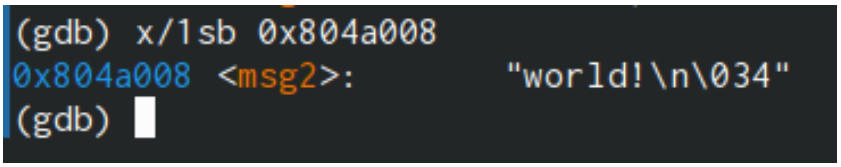

Рис. 4.28: Просмотр значения переменной msg2 по адресу

Изменим первый символ переменной msg1 (рис. 4.29).

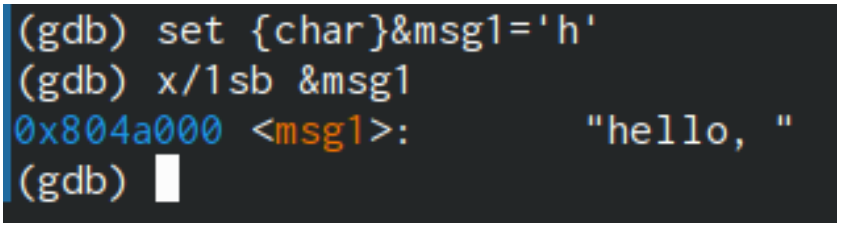

Рис. 4.29: Изменение первого символа переменной msg1

Изменим первый символ переменной msg2 (рис. 4.30).

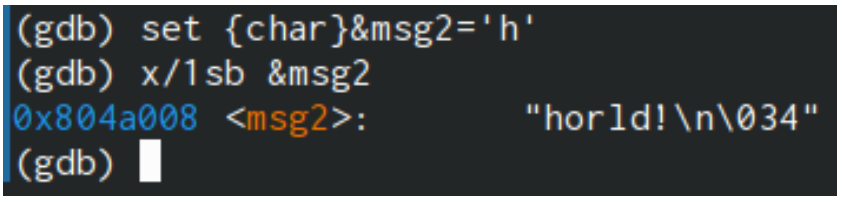

Рис. 4.30: Изменение первого символа переменной msg2

Выведем в шестнадцатеричном формате, в двоичном формате и в символьном виде значение регистра edx (рис. 4.31).

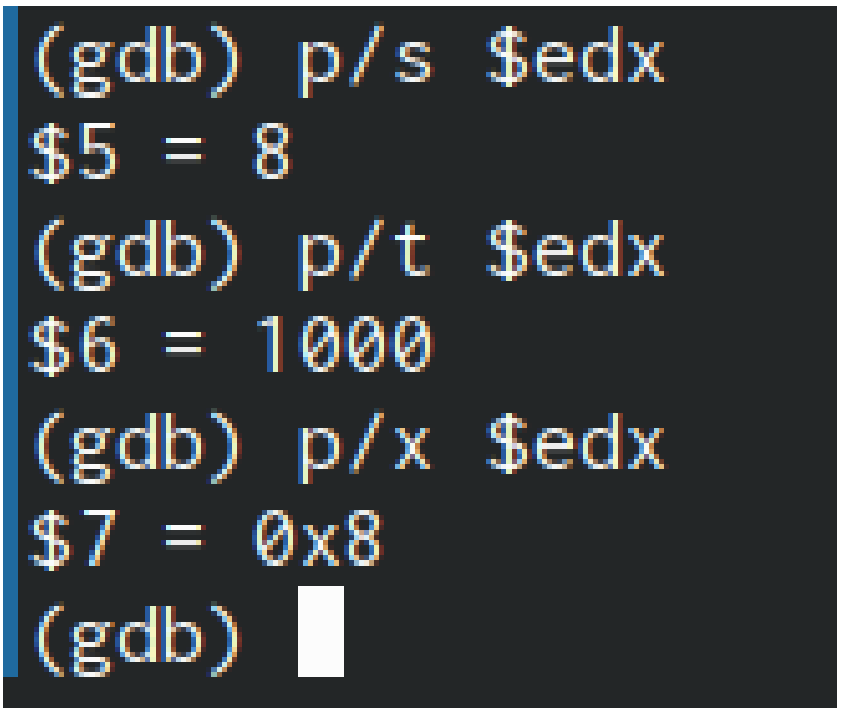

Рис. 4.31: Вывод в различных форматах значения регистра edx

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. 4.32).

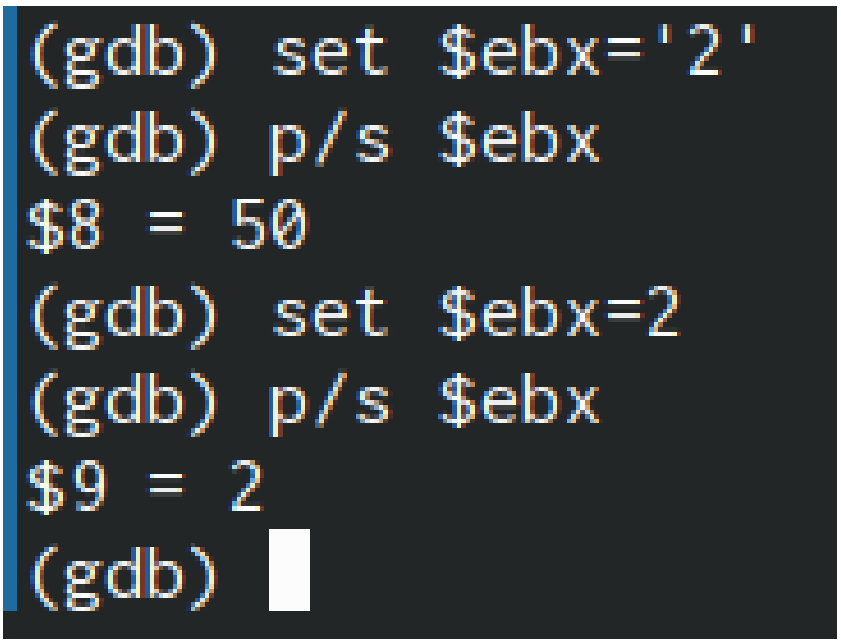

Рис. 4.32: Изменение значения регистра ebx

Завершим выполнение программы с помощью команды stepi (сокращенно si) и выйдем из GDB с помощью команды quit (сокращенно q) (рис. 4.33).

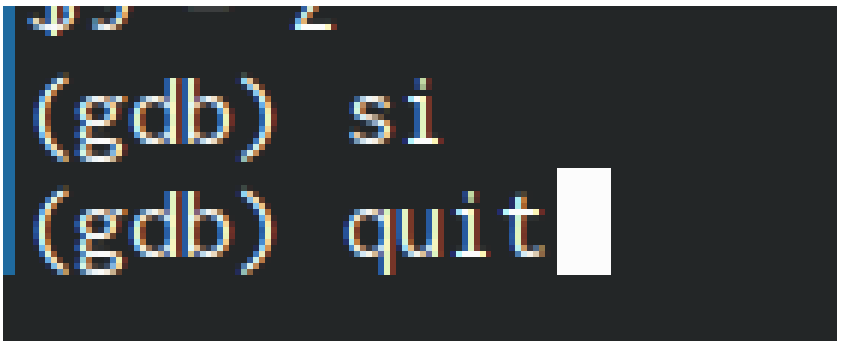

Рис. 4.33: Завершение выполнения программы и выход из GDB

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. 4.34).

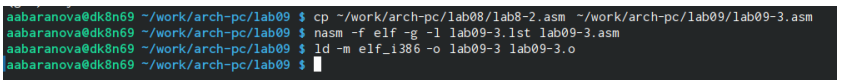

Рис. 4.34: Копирование файла lab8-2.asm в файл с именем lab09-3.asm и создание исполняемого файла

Загрузим исполняемый файл в отладчик, указав аргументы (рис. 4.35).

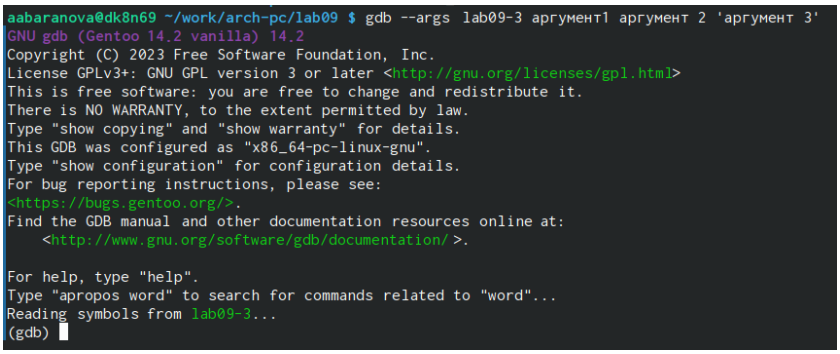

Рис. 4.35: Загрузка исполняемого файла в отладчик

Установим точку останова перед первой инструкцией в программе и запустим её (рис. 4.36).

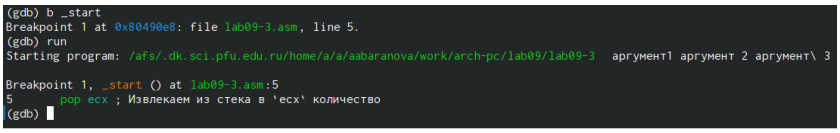

Рис. 4.36: Установка точки останова перед первой инструкцией в программе и её запуск

Проверим адрес вершины стека и убедимся, что там хранится 5 элементов (рис. 4.37).

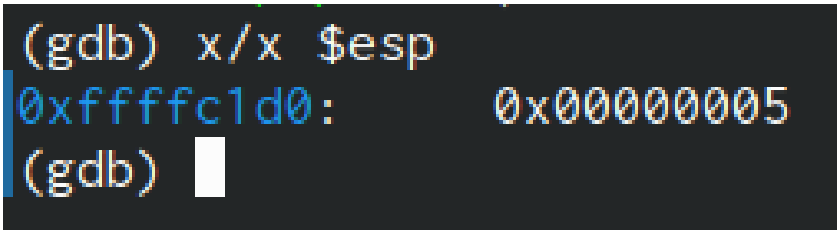

Рис. 4.37: Проверка адреса вершины стека

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. 4.38).

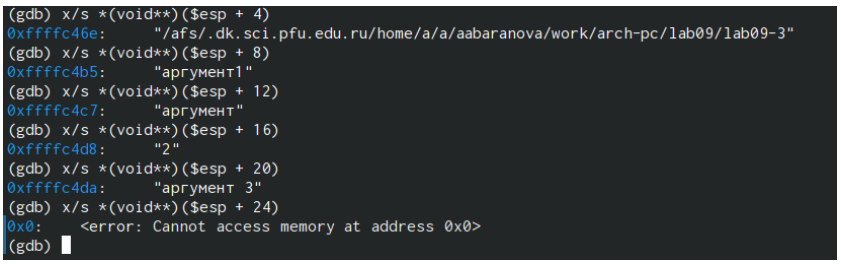

Рис. 4.38: Просмотр всех позиций стека

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. 4.39), (рис. 4.40), (рис. 4.41), (рис. 4.42).

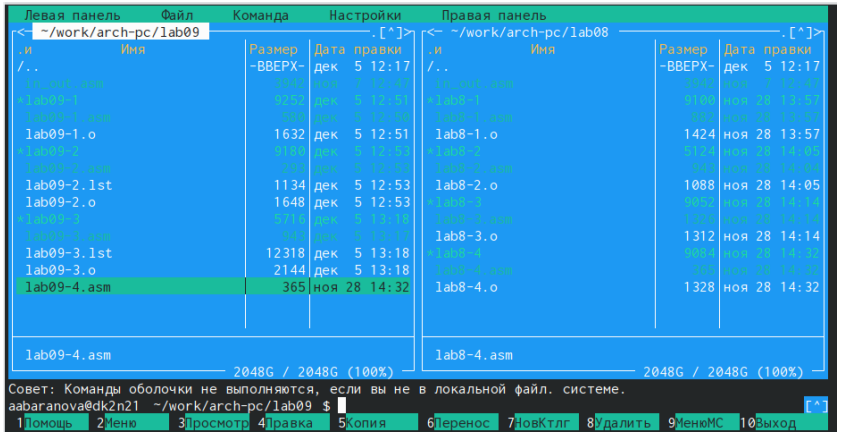

Рис. 4.39: Преобразование программы из лабораторной работы №8

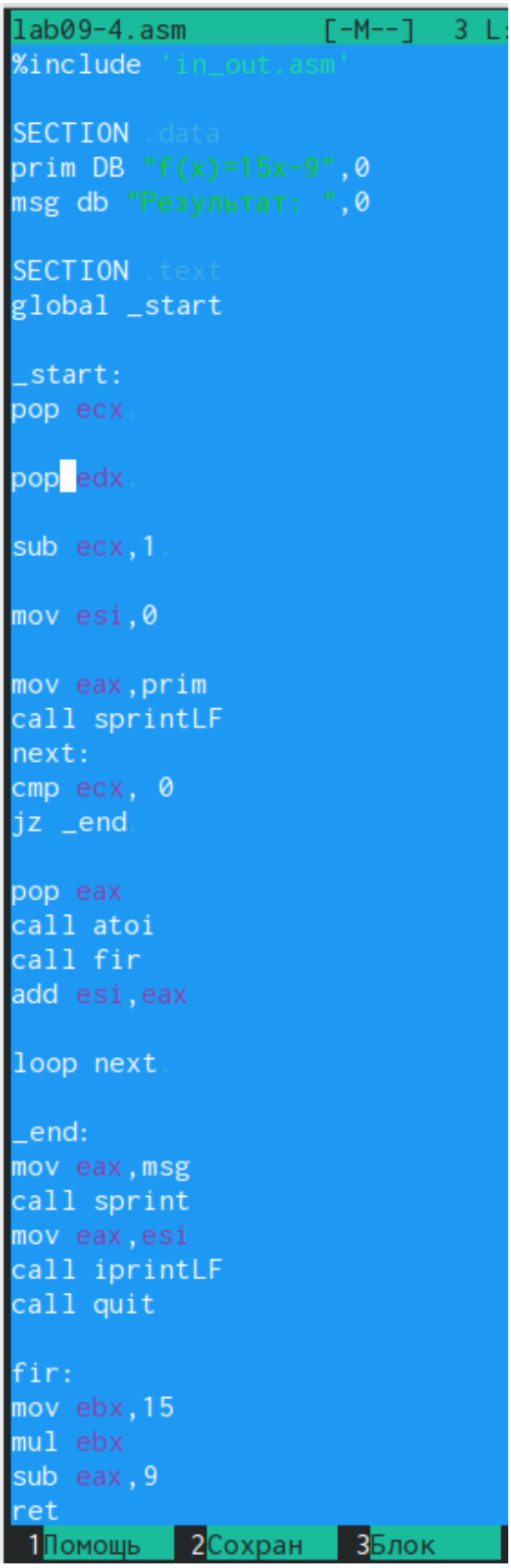

Рис. 4.40: Преобразование программы из лабораторной работы №8

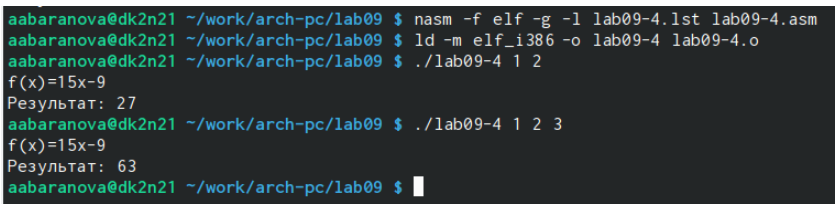

Рис. 4.41: Преобразование программы из лабораторной работы №8

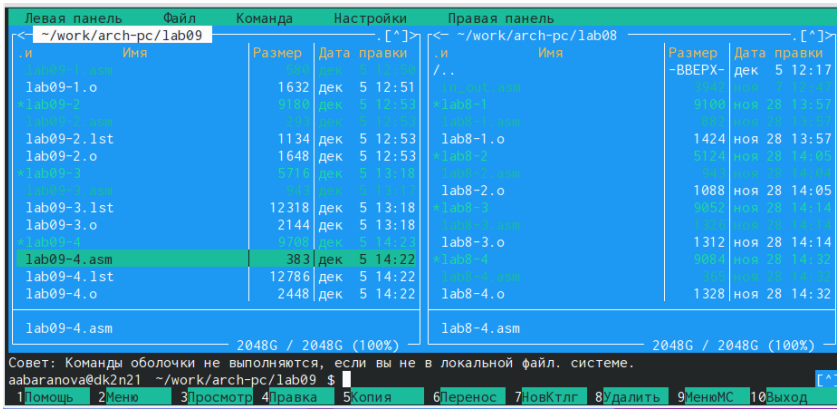

Рис. 4.42: Преобразование программы из лабораторной работы №8

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. 4.43), (рис. 4.44), (рис. 4.45), (рис. 4.46), рис. 4.47), рис. 4.48), рис. 4.49), рис. 4.50), рис. 4.51), рис. 4.52).

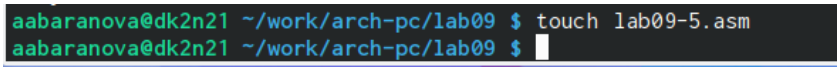

Рис. 4.43: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

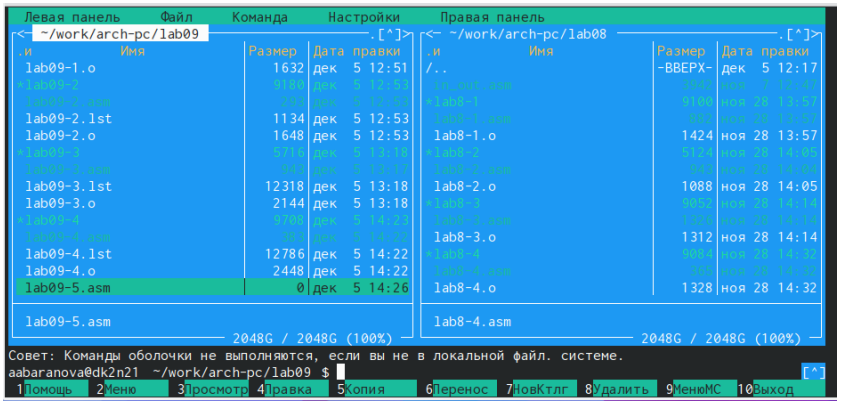

Рис. 4.44: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

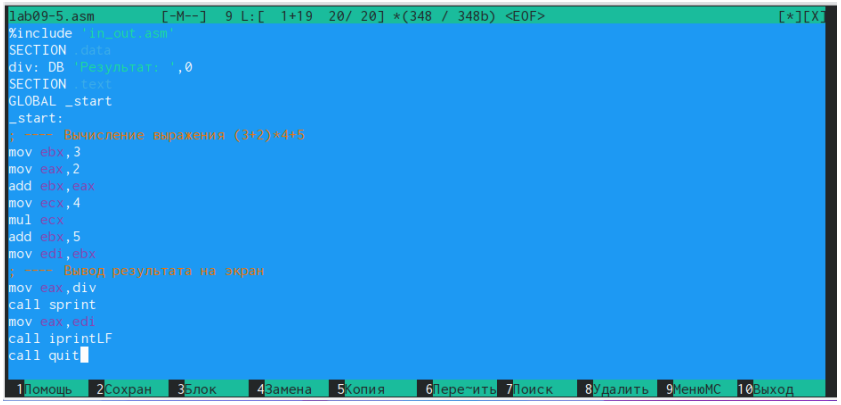

Рис. 4.45: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

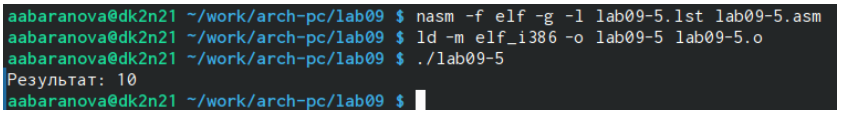

Рис. 4.46: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

Рис. 4.47: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

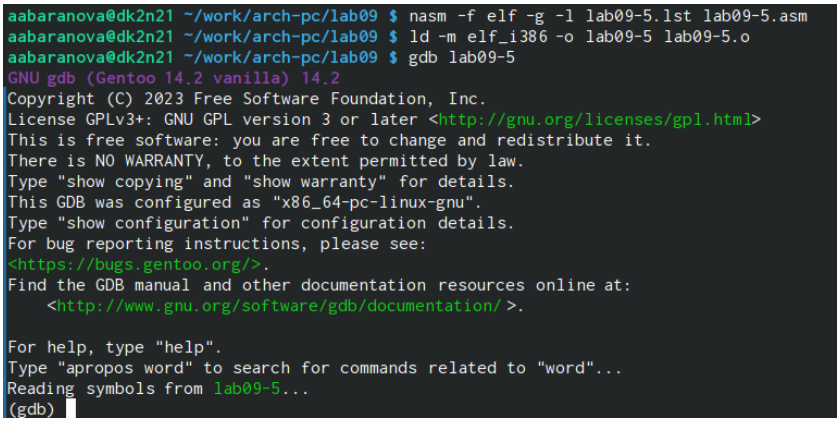

Рис. 4.48: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

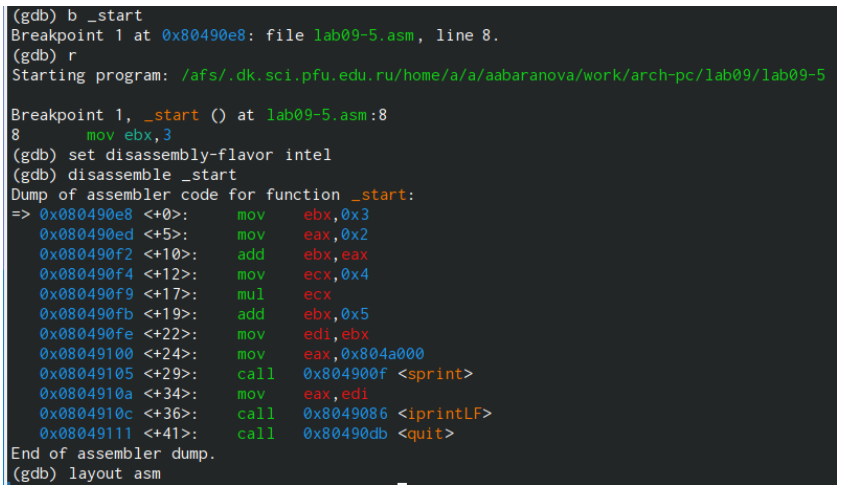

Рис. 4.49: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

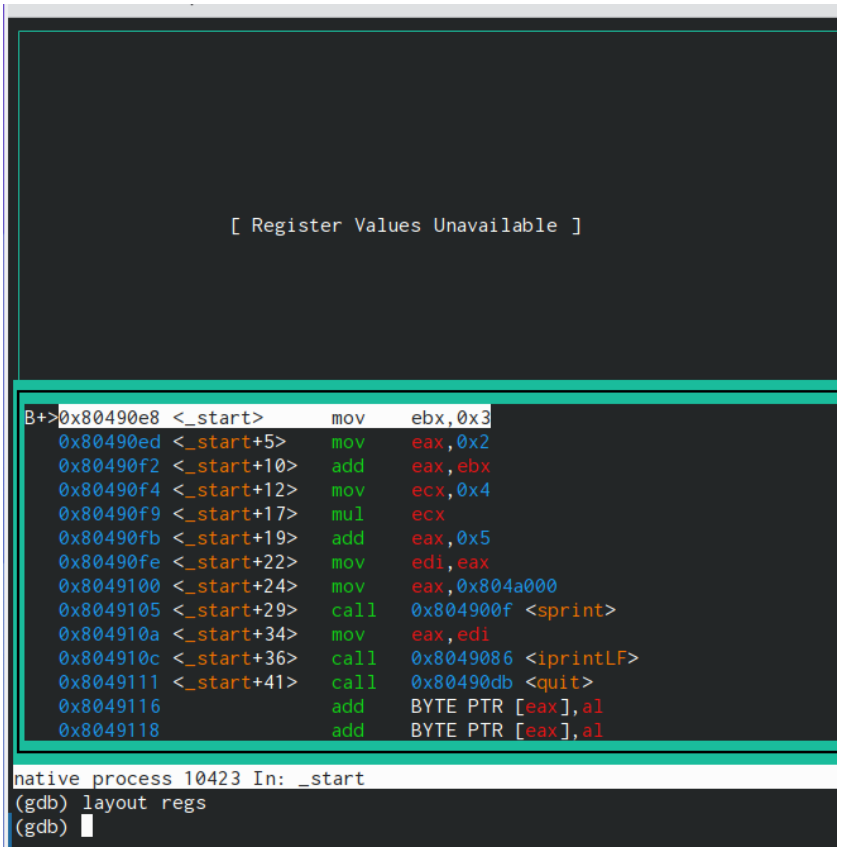

Рис. 4.50: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

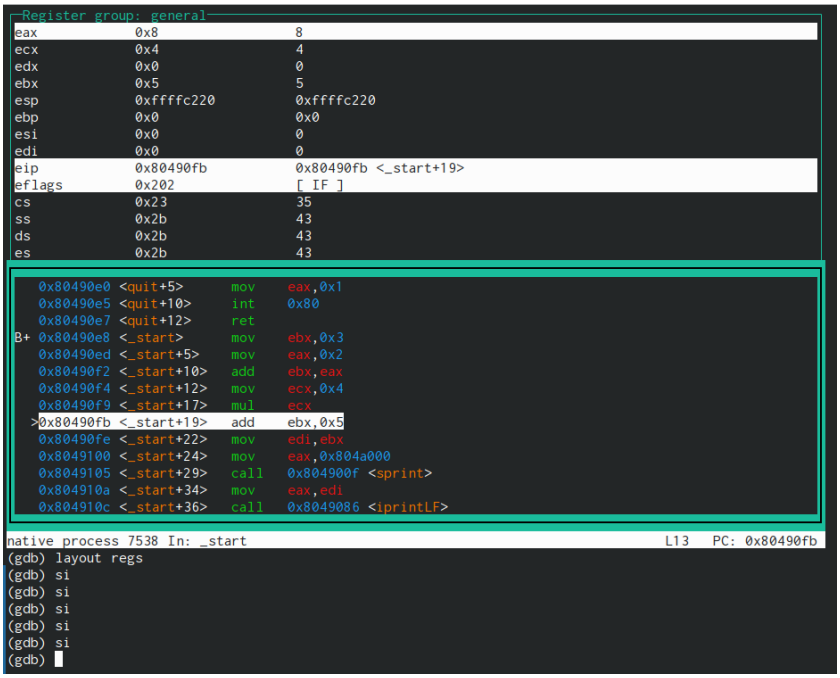

Рис. 4.51: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

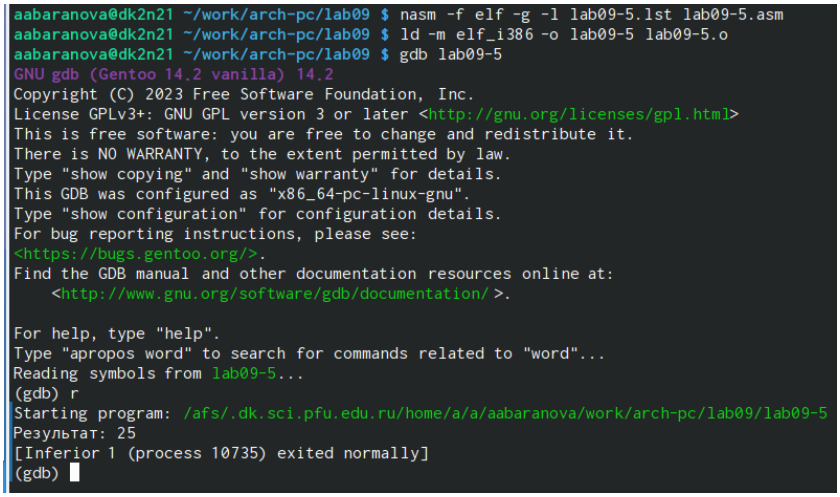

Рис. 4.52: Обнаружение ошибки в программе с помощью отладчика GDB и её исправление

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.