Отчёта по лабораторной работе №9

Понятие подпрограммы. Отладчик GDB.

Сибомана Ламек НКАбд-03-24

Содержание

# 1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

# 2 Теоретическое введение

### 2.0.1 Понятие об отладке

**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. |

### 2.0.2 Методы отладки

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: • 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 програм- мы, т.е. они сохраняются до выхода из программы-отладчика или до смены отлаживаемой программы.

# 3 Выполнение лабораторной работы

## 3.1 Реализация подпрограмм в NASM

1.Сначала я создал каталог для программам лабораторной работы №9, затем перешёл в него и создал файл lab09-1.asm (рис. fig. **¿fig:001?**)

![Создание каталога и файла lab09-1.asm] (image/1.png){#fig:001 width=70%}

Открывал файл в Midnight Commander и заполняем его в соответствии с листингом 9.1 (рис. fig. **¿fig:002?**).

![Текст программы lab09-1.asm] (image/2.png){#fig:002 width=70%}

Создал исполняемый файл и проверьте его работу.

![Создание и запуск lab09-1.asm]image/3.png){#fig:003 width=70%}

Изменил текст программы добавив изменение значение регистра ecx в циклеСнова открывал файл для редактирования и изменяем его, добавив изменение значения регистра в цикле (рис. fig. 1)

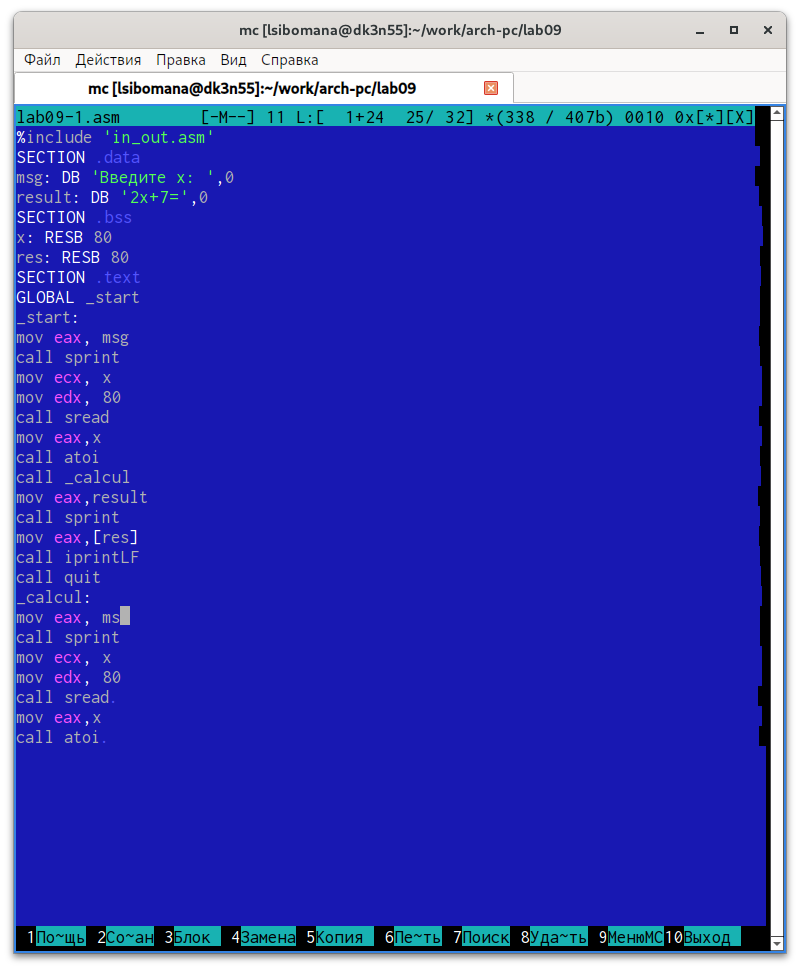


Рис. 1: Текст программы lab09-1.asm

Создаем исполняемый файл и запускаем его (рис. fig. **¿fig:005?**).

![Создание и запуск lab09-1.asm]image/5.png){#fig:005 width=70%}

## 3.2 Отладка программам с помощью GDB

Создаем новый файл в каталоге(рис. fig. **¿fig:006?**). Создание и запуск lab09-1.asm

Создаем исполняемый файл и запускаем его

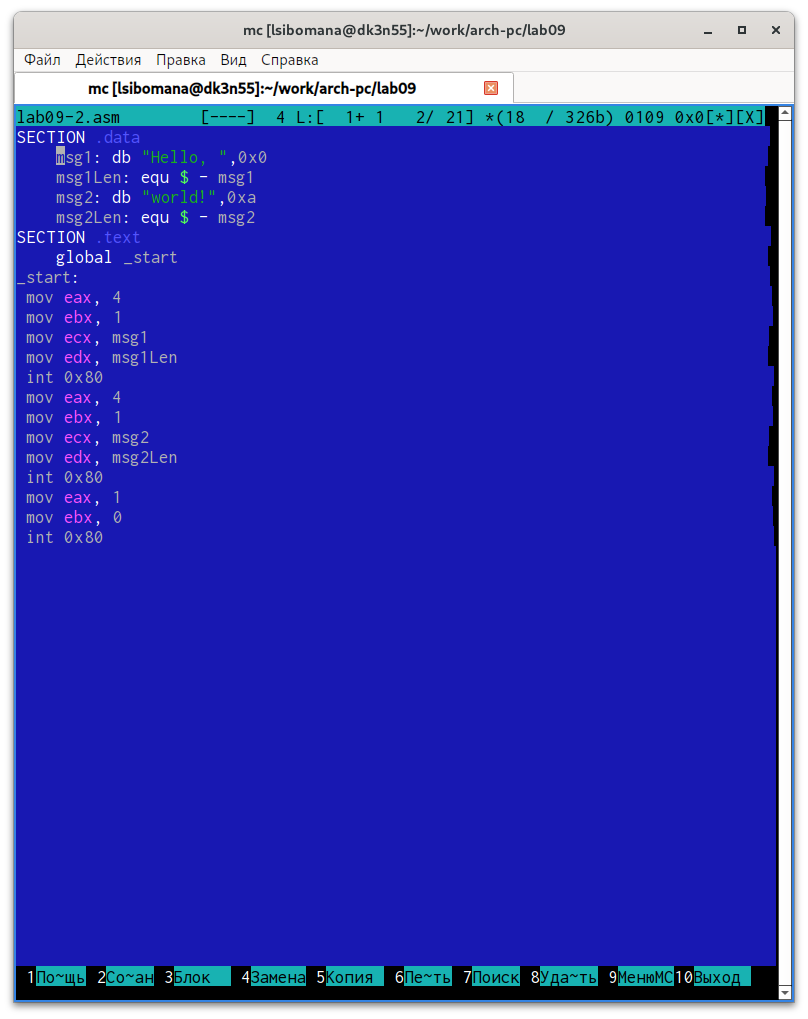


Рис. 2: Текст программы lab09-1.asm

Получаем исходный файл с использованием отладчика gdb

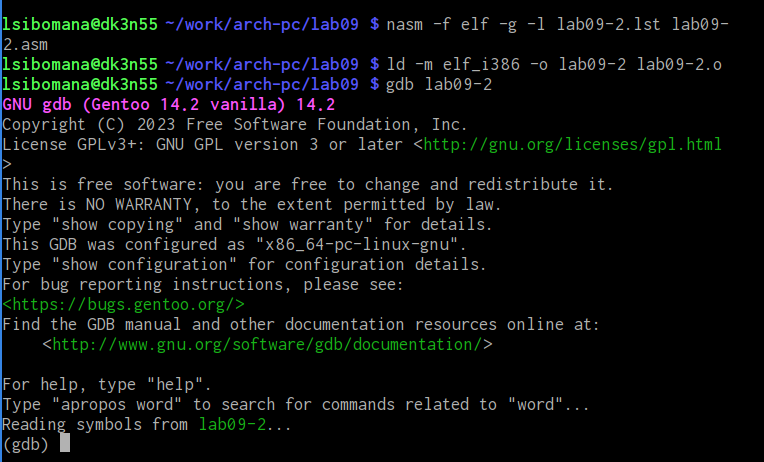


Рис. 3: Создание lab09-2.asm

Запускаем команду в отладчике

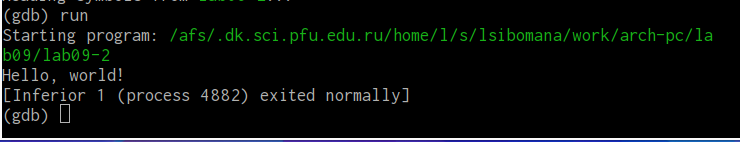


Рис. 4: Текст программы lab09-2.asm

Устанавливаем брейкпоинт на метку \_start и запускаем программу

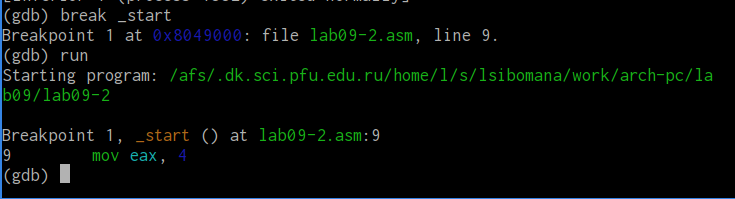


Рис. 5: Создание и запуск lab09-2.asm

Смотрим дисассимилированный код программы с помощью команды disassemble, начиная с метки \_start



Рис. 6: Создание

Переключаемся на отображение команд с Intel’овским синтаксисом (рис. fig. **¿fig:012?**).



Рис. 7: Текст программы

Различия отображения синтаксиса машинных команд в режимах ATT и Intel:

1.Порядок операндов: В ATT синтаксисе порядок операндов обратный, сначала указывается исходный операнд, а затем - результирующий операнд. В Intel синтаксисе порядок обычно прямой, результирующий операнд указывается первым, а исходный - вторым.

2.Разделители: В ATT синтаксисе разделители операндов - запятые. В Intel синтаксисе разделители могут быть запятые или косые черты (/).

3.Префиксы размера операндов: В ATT синтаксисе размер операнда указывается перед операндом с использованием префиксов, таких как “b” (byte), “w” (word), “l” (long) и “q” (quadword). В Intel синтаксисе размер операнда указывается после операнда с использованием суффиксов, таких как “b”, “w”, “d” и “q”.

4.Знак операндов: В ATT синтаксисе операнды с позитивными значениями предваряются символом “$". В Intel синтаксисе операнды с позитивными значениями могут быть указаны без символа "$”.

5.Обозначение адресов: В ATT синтаксисе адреса указываются в круглых скобках. В Intel синтаксисе адреса указываются без скобок.

6.Обозначение регистров: В ATT синтаксисе обозначение регистра начинается с символа “%”. В Intel синтаксисе обозначение регистра может начинаться с символа “R” или “E” (например, “%eax” или “RAX”).

Включаем режим псевдографики (рис. fig. **¿fig:013?**).

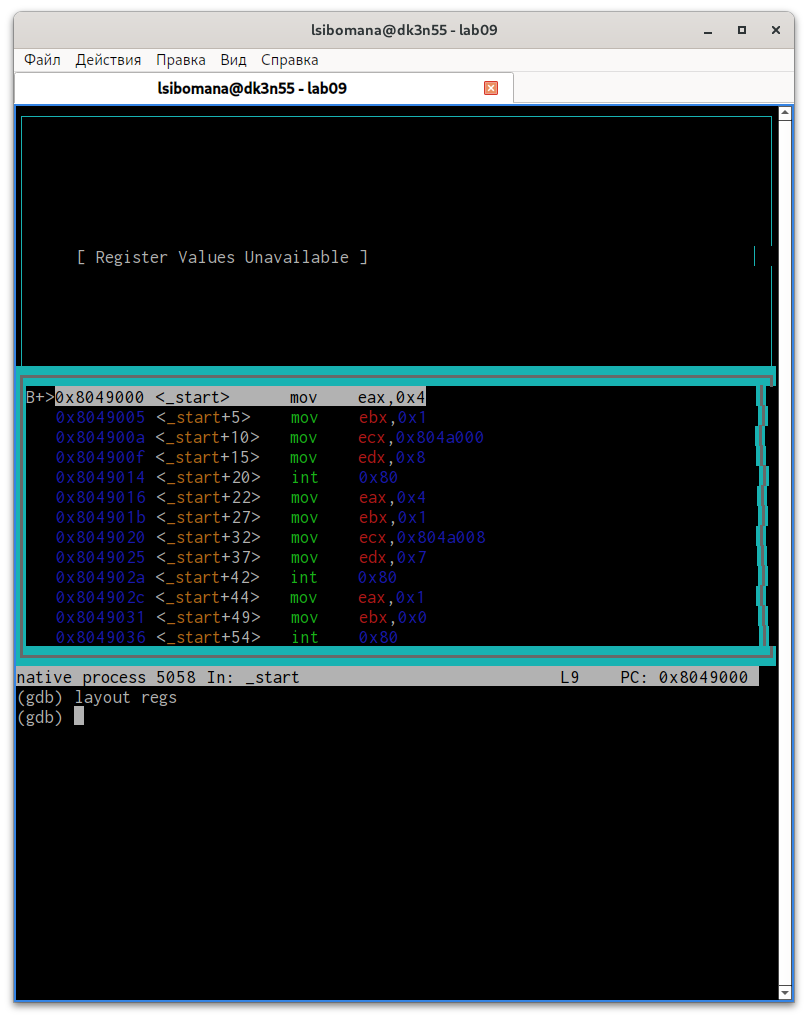


Рис. 8: Создание

Проверяем была ли установлена точка останова и устанавливаем точку останова предпоследней инструкции (рис. fig. **¿fig:014?**).

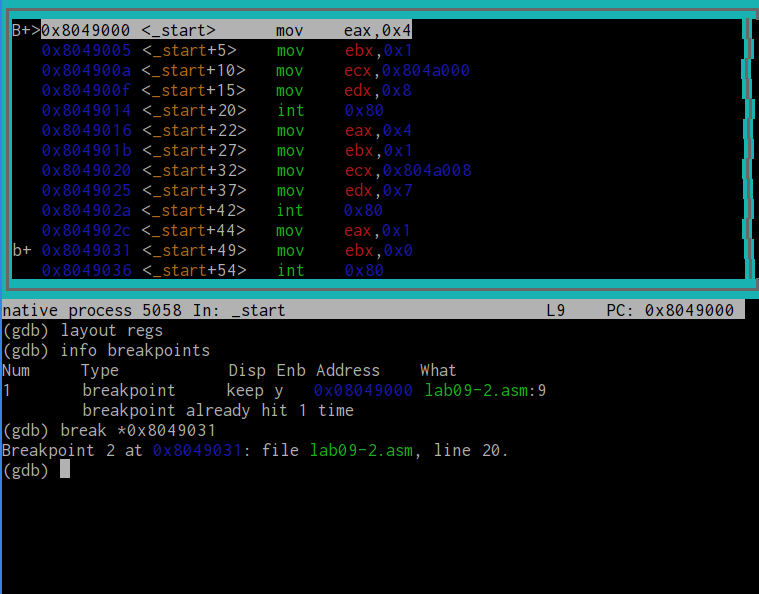


Рис. 9: Текст программы

Посмотрим информацию о всех установленных точках останова (рис. fig. **¿fig:015?**).

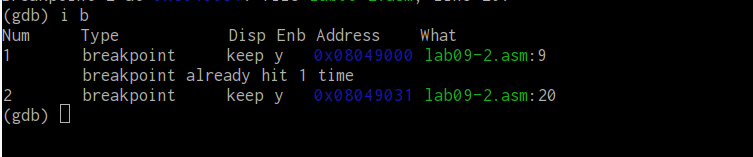


Рис. 10: layout and info

Выполняем 5 инструкций командой si (рис. fig. **¿fig:016?**).

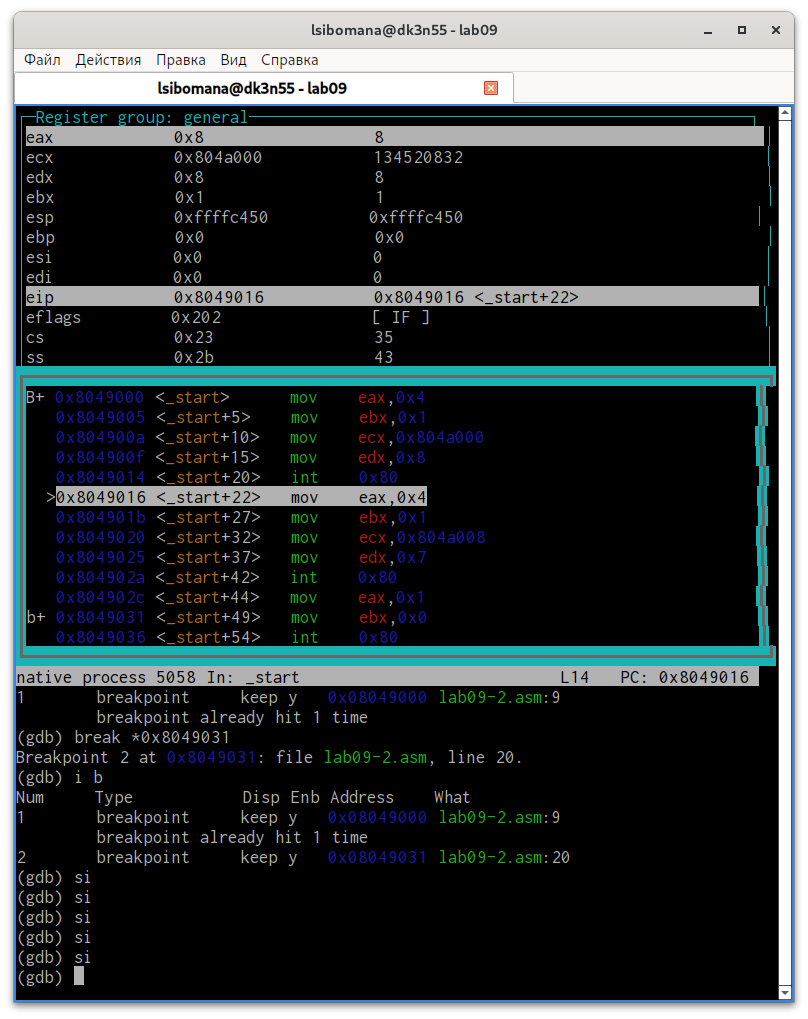
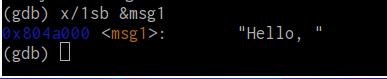


Рис. 11: текст

Во время выполнения команд менялись регистры: ebx, ecx, edx,eax, eip.

Смотрим значение переменной msg1 по имени (рис. fig. **¿fig:017?**).

{#fig:0017width=70%}

Смотрим значение переменной msg2 по адресу (рис. fig. **¿fig:018?**).

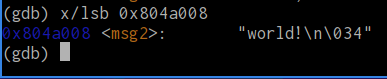


Рис. 12: Сx/lsb

Изменим первый символ переменной msg1 (рис. fig. **¿fig:019?**).

Меняем символ

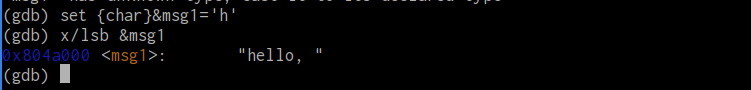


Рис. 13: change of code

Изменим первый символ переменной msg2 (рис. fig. **¿fig:020?**).

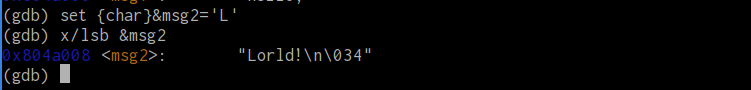
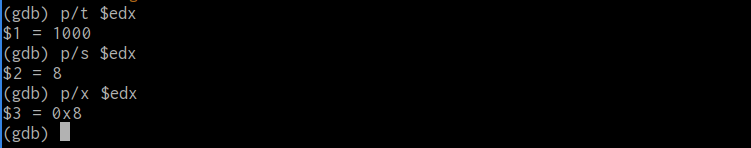


Рис. 14: change of code

Смотрим значение регистра edx в разных форматах (рис. fig. **¿fig:021?**). 

Изменяем регистор ebx (рис. fig. **¿fig:022?**)

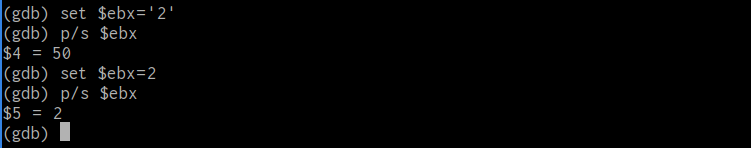
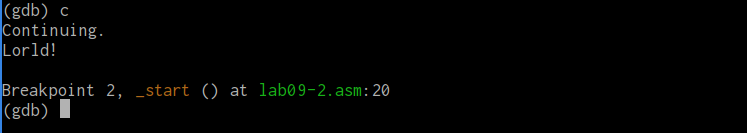


Рис. 15: change of code

Выводится разные значения, так как команда без кеавычек присваивает регистру вводимое значение.

Прописываем команды для завершения программы и выхода из GDB (рис. fig. **¿fig:023?**). 

Копируем файл lab8-2.asm в файл с именем lab09-3.asm (рис. fig. **¿fig:024?**)

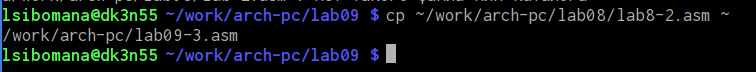


Рис. 16: change of code

Создаем исполняемый файл и запускаем его в отладчике GDB (рис. fig. **¿fig:025?**).

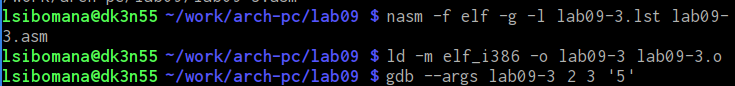
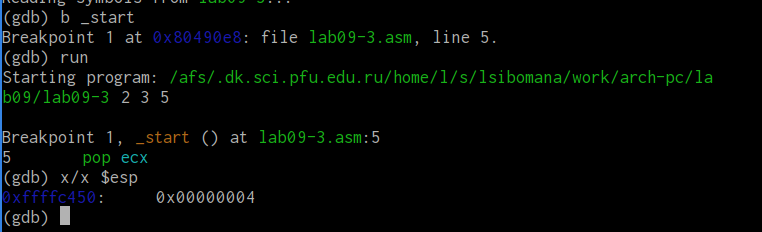


Рис. 17: change of code

Установим точку останова перед первой инструкцией в программе и запустим ее (рис. fig. **¿fig:026?**).  Смотрим позиции стека по разным адресам (рис. fig. **¿fig:027?**).

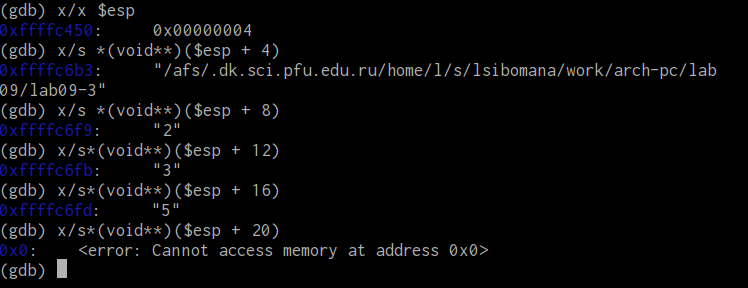


Рис. 18: change of code

Шаг изменения адреса равен 4 потому что адресные регистры имеют размерность 32 бита(4 байта).

## 3.3 Задание для самостоятельной работы

### 3.3.1 Задание 1

Копируем файл lab8-4.asm(ср №1 в ЛБ8) в файл с именем lab09-3.asm (рис. fig. **¿fig:028?**).

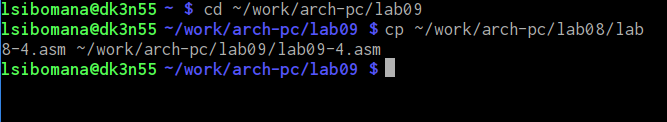


Рис. 19: change of code

Открываем файл в Midnight Commander и меняем его, создавая подпрограмму (рис. fig. **¿fig:029?**).

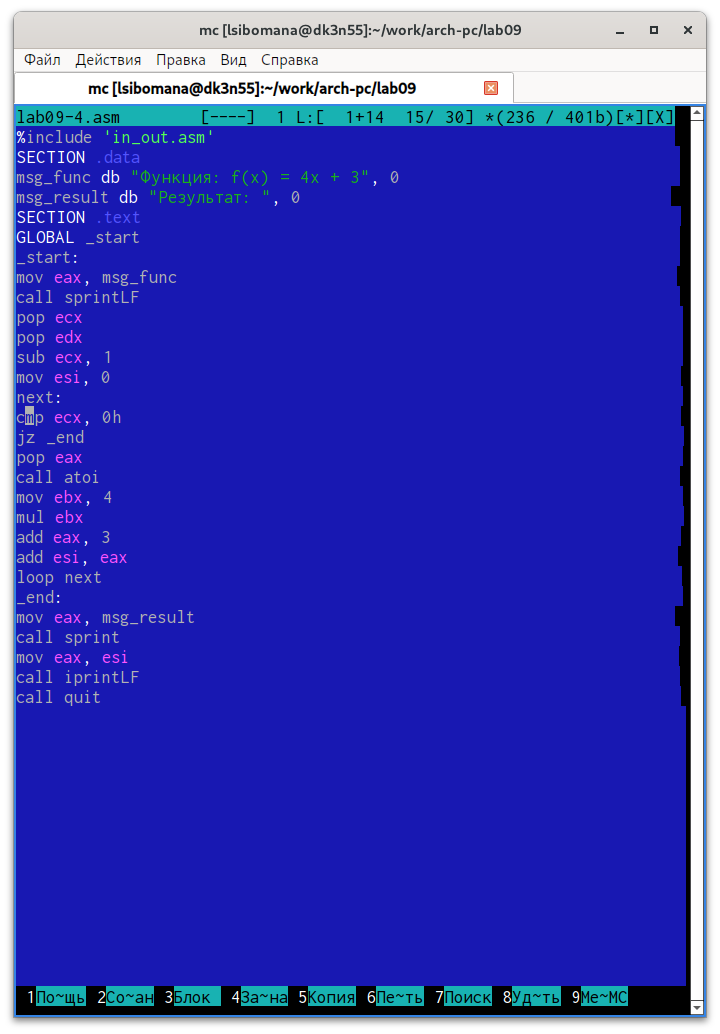
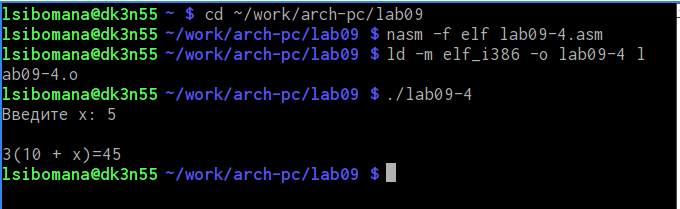


Рис. 20: Текст программы

Создаем исполняемый файл и запускаем его (рис. fig. **¿fig:030?**).

 ### Задание 2

Создаем новый файл в дирректории (рис. fig. **¿fig:031?**).

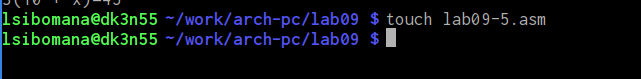


Рис. 21: change of code

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 9.3 (рис. fig. **¿fig:032?**).

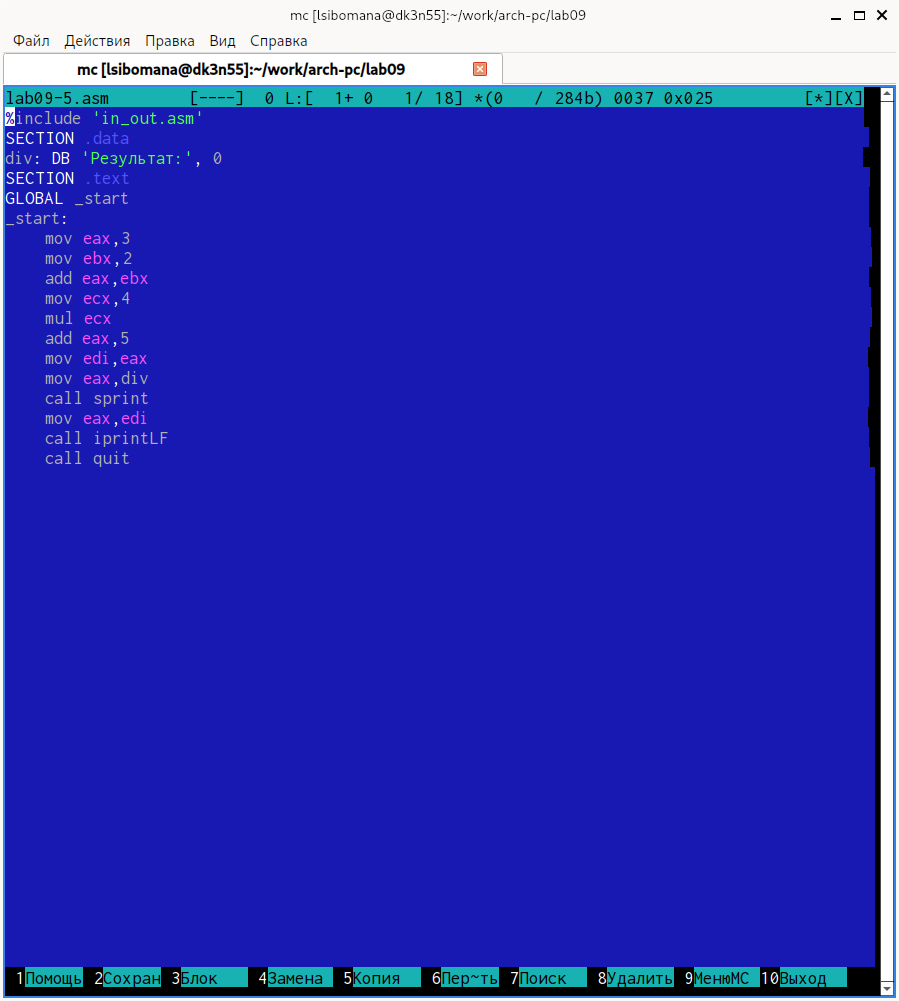


Рис. 22: Текст программы

Создаем исполняемый файл и запускаем его (рис. fig. **¿fig:033?**).

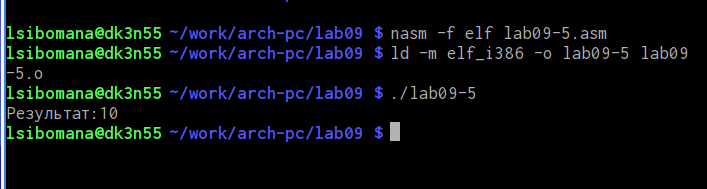
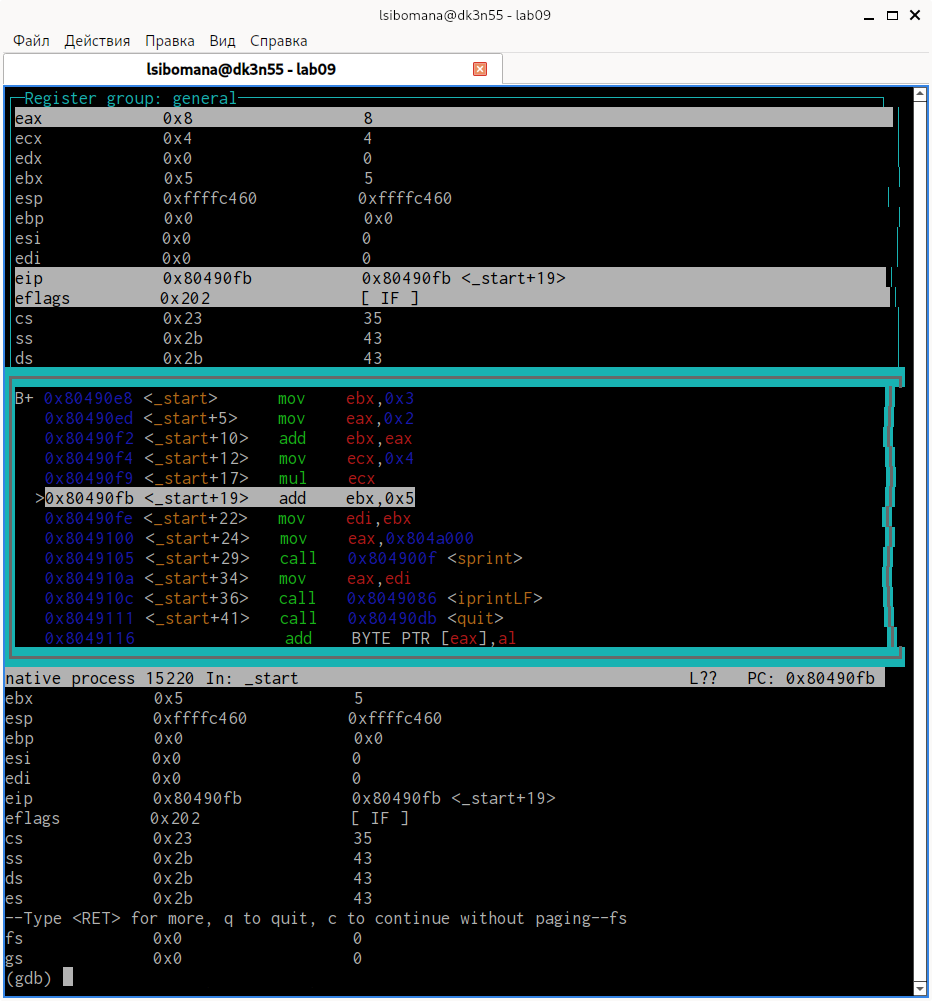


Рис. 23: Создание и запуск lab09-5.asm

Создаем исполняемый файл и запускаем его в отладчике GDB и смотрим на изменение решистров командой si (рис. fig. **¿fig:034?**). 

Изменяем программу для корректной работы (рис. fig. **¿fig:035?**).

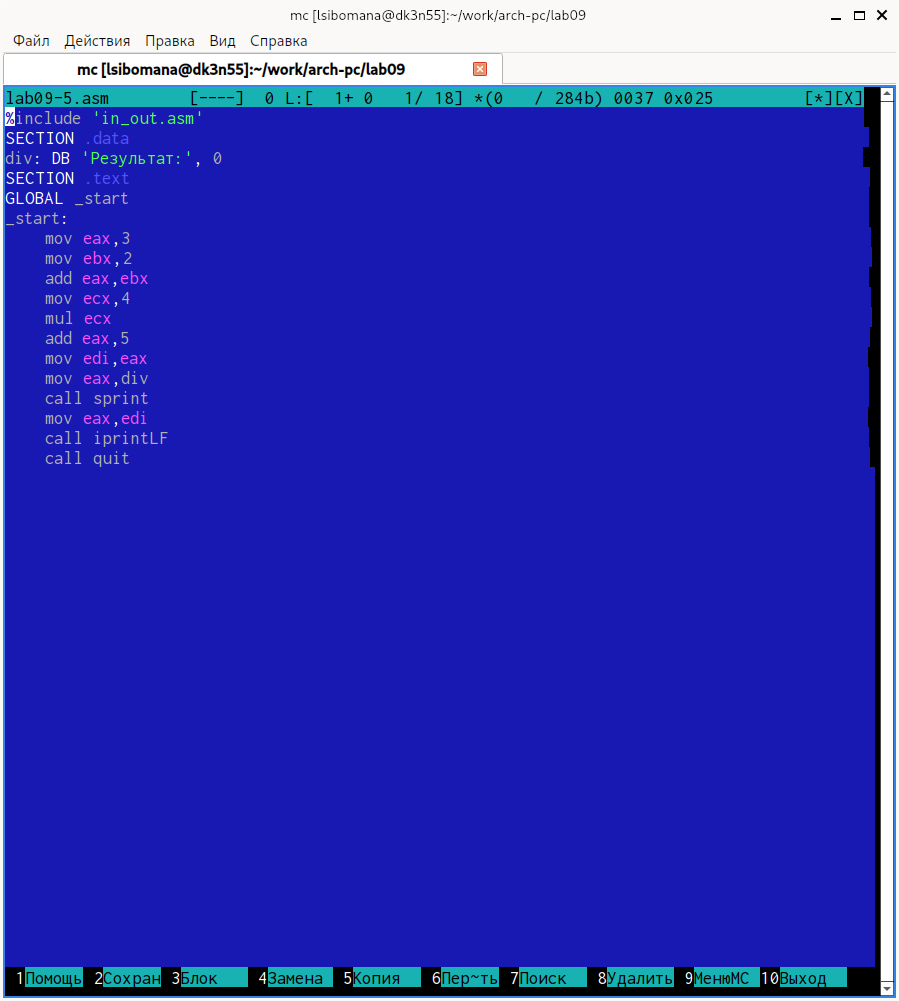


Рис. 24: Текст программы

Создаем исполняемый файл и запускаем его (рис. fig. **¿fig:036?**).

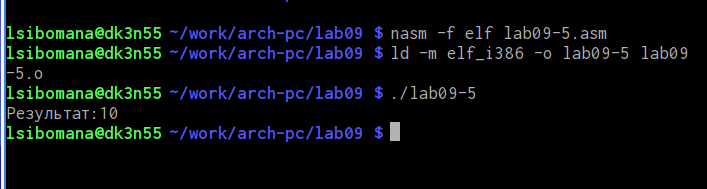


Рис. 25: Создание и запуск lab09-5.asm

# 4 Выводы

Мы познакомились с методами отладки при помощи GDB и его возможностями.