Отчёт по лабораторной работе №9

Дисциплина: архитектура компьютера

Кузнецова Елизавета Андреевна

Содержание

# 1 Цель работы

Освоить работу с подпрограммами и отладчиком gdb.

# 2 Задание

1. Ознакомиться с листингами и использовать их при работе gdb.
2. Выполнить задания для самостоятельной работы.
3. Загрузить файлы на Github.

# 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 сообщения); использование специальных программ-отладчиков. GDB (GNU Debugger — отладчик проекта GNU) работает на многих UNIX-подобных системах и умеет производить отладку многих языков программирования. 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.

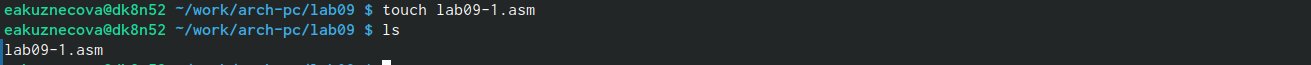
# 4 Выполнение лабораторной работы

С помощью утилиты mkdir создала директорию, в которой буду создавать файлы с программами для лабораторной работы. Перешла в созданный каталог с помощью утилиты cd (рис. [??]).

Создание директории

Создание директории

С помощью утилиты touch создала файл lab09-1.asm (рис. [??]).



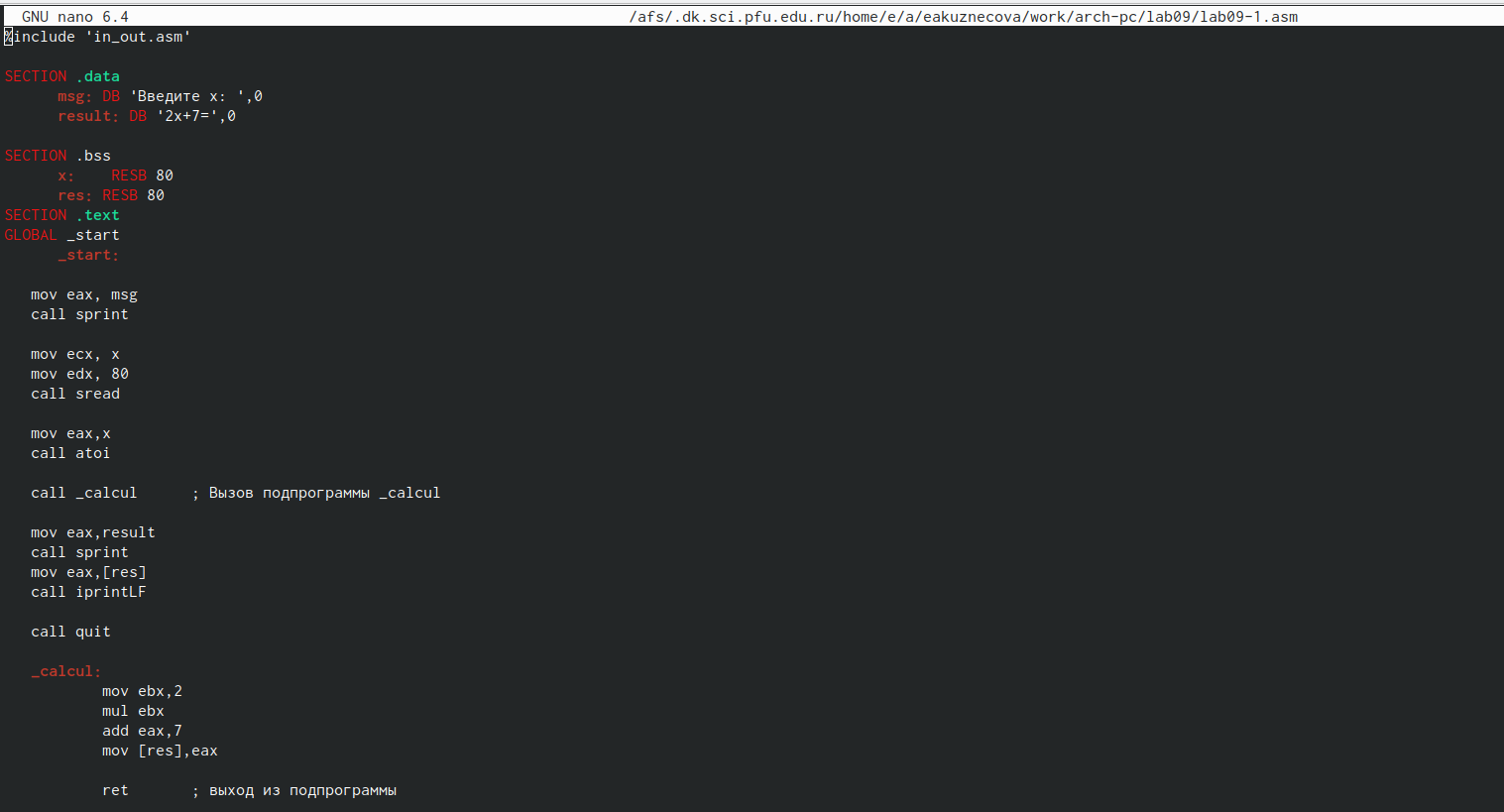
Создание файла

Скопировала в текущий каталог файл in\_out.asm с помощью утилиты cp, так как он будет использоваться в других программах (рис. [??]).

Копирование файла

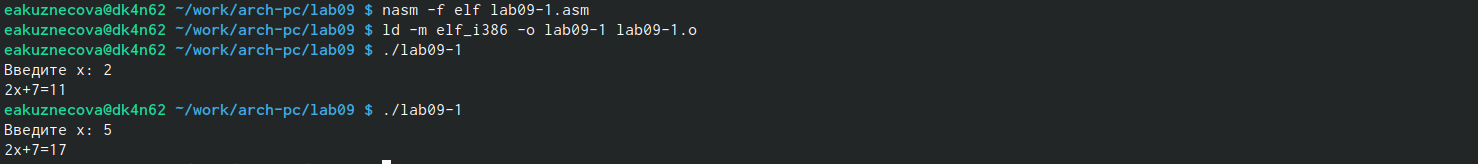
Копирование файла

Открыла созданный файл lab09-1.asm, вставила в него программу (рис. [??]).



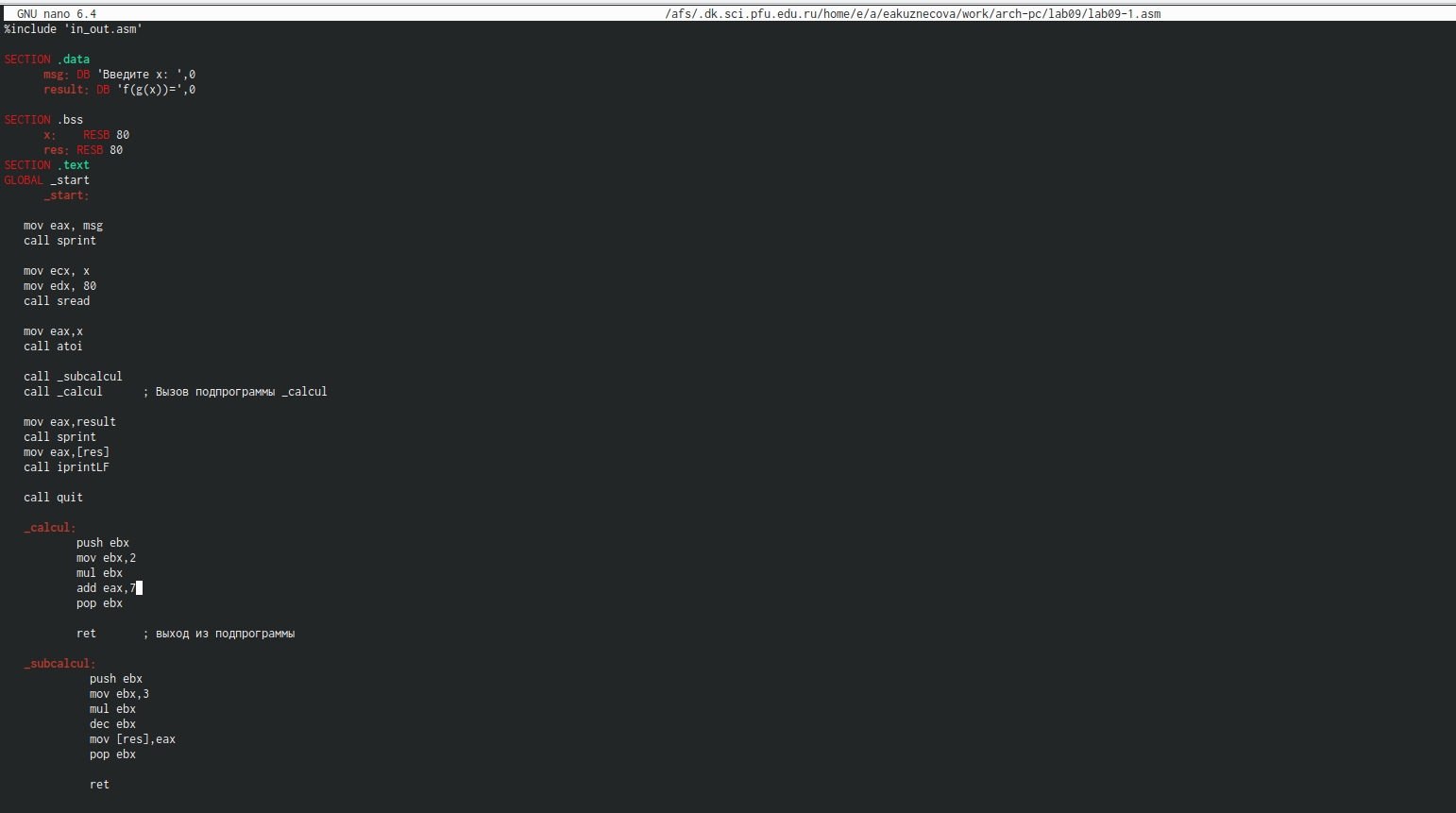
Редактирование файла

Создала исполняемый файл программы и запустила его. Результат данной программы являлся правильным (рис. [??]).



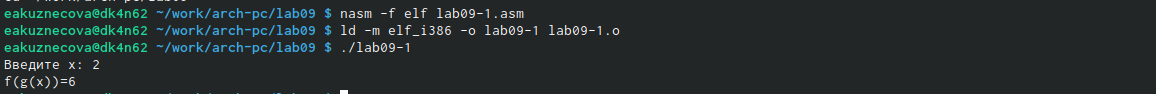
Запуск исполняемого файла

Изменила текст программы, добавив подпрограмму \_subcalcul в подрограмму \_calcul (рис. [??]).



Редактирование файла

Создала новый исполняемый файл программы и запустила его. Результат данной программы являлся правильным (рис. [??]).



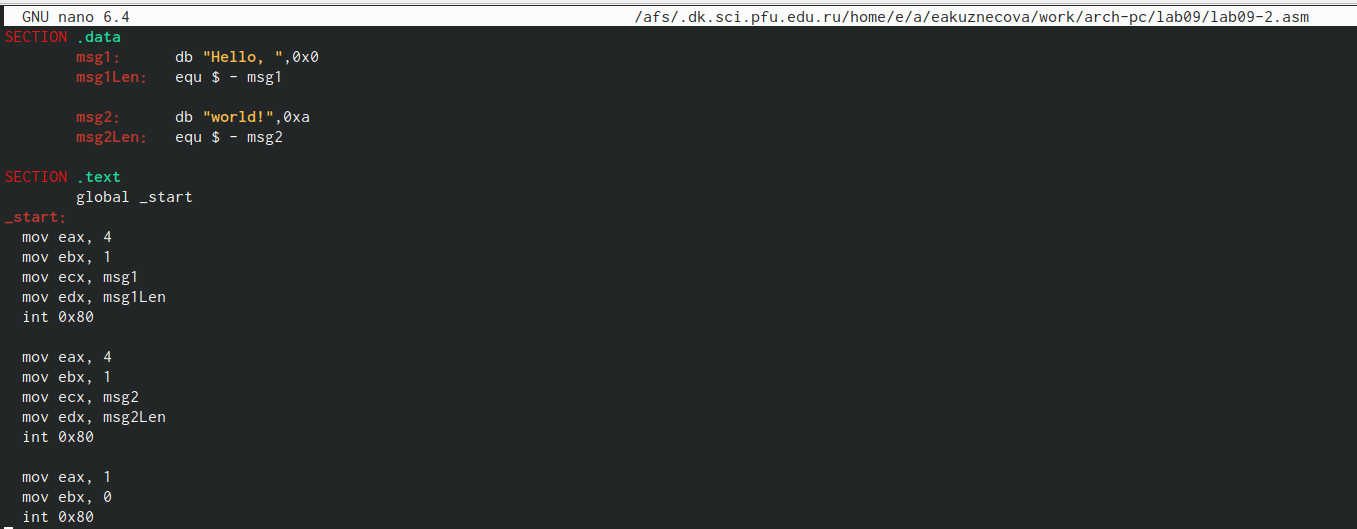
Запуск исполняемого файла

С помощью утилиты touch создала файл lab09-2.asm (рис. [??]).

Создание файла

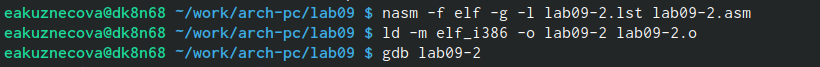
Создание файла

Открыла созданный файл lab09-2.asm, вставила в него программу (рис. [??]).



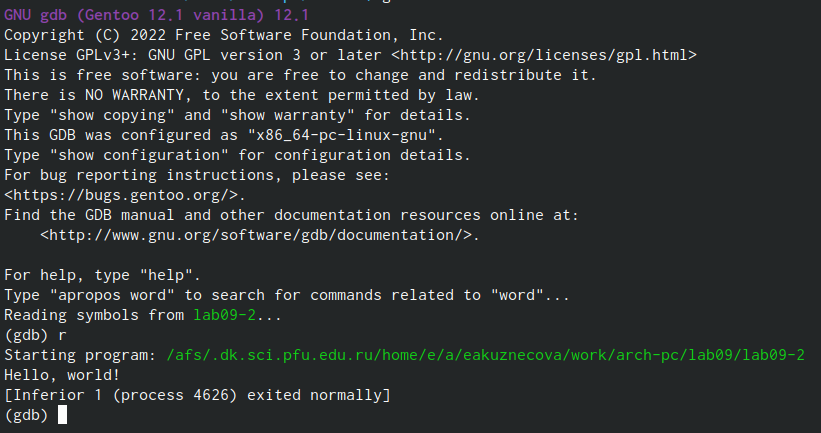
Редактирование файла

Создала новый исполняемый файл программы. Для работы с GDB в исполняемый файл добавила отладочную информацию, провела трансляцию программ с ключом ‘-g’. Загрузила исполняемый файл в отладчик gdb (рис. [??]).



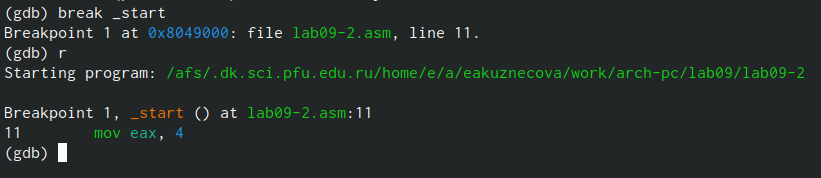
Загрузка исполняемого файла в gdb

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



Проверка работы программы

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



Запуск программы

Посмотрела дисассимилированный код программы с помощью команды disassemble, начиная с метки \_start (рис. [??]).



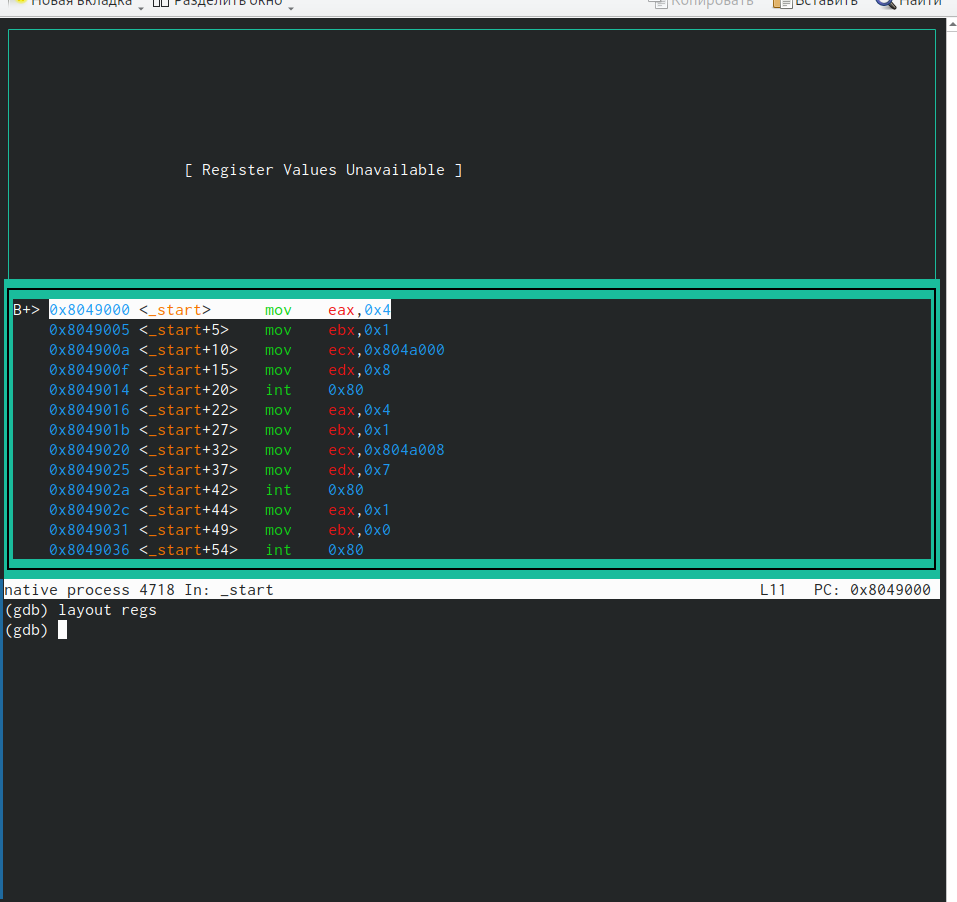
Просмотр дисассимилированного кода программы

Переключилась на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel (рис. [??]). В представлении ATT в виде 16-ричного числа записаны первые аргументы всех команд, а в представлении intel так записываются адреса вторых аргументов.



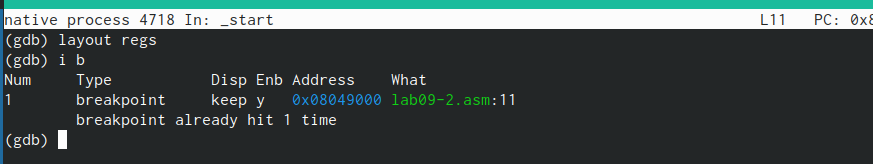
Просмотр дисассимилированного кода программы в режиме intel

Включила режим псевдографики для более удобного анализа программы (рис. [??]).



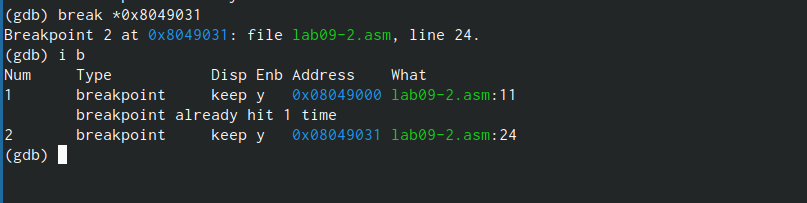
Режим псевдографики

Посмотрела информацию о наших точках останова. Сделала это с помощью команды info breakpoints (кратко i b) (рис. [??]).



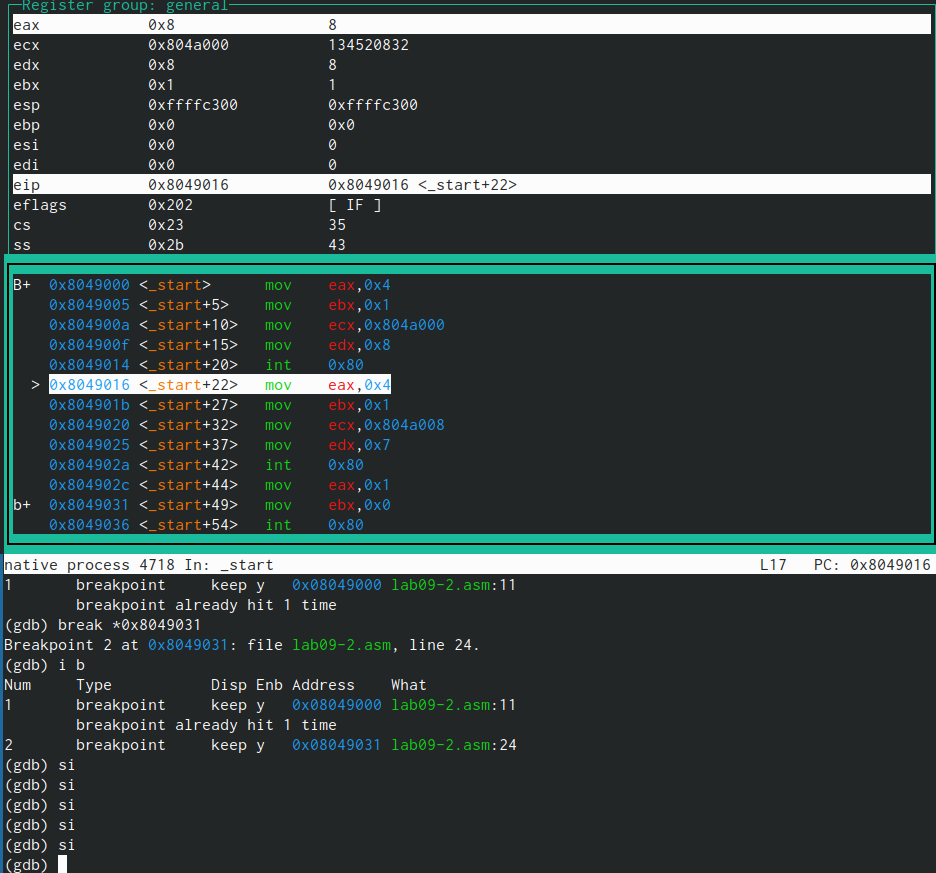
Просмотр точек останова

Добавила еще одну точку останова по адресу инструкции. Посмотрела информацию о всех установленных точках останова (рис. [??]).



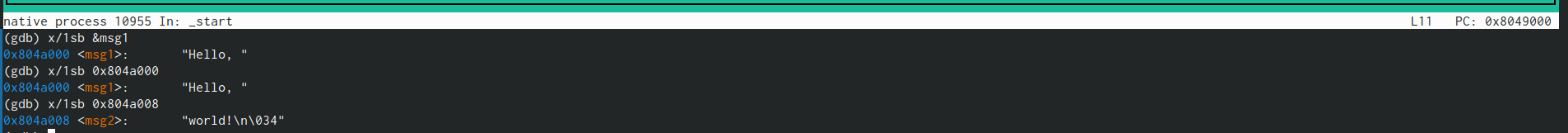
Добавление точки останова

Выполнила 5 инструкций с помощью команды stepi и проследила за изменением значений регистров. Значение регистров eax,ecx,edx,ebx (рис. [??]).



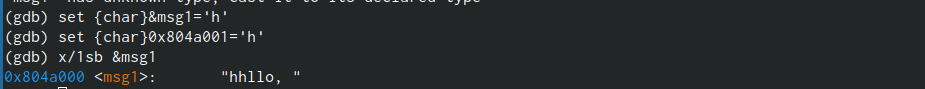
Изменение значений регистров

Вывела текущее значение переменных (msg1 и msg2) по имени и по адресу (рис. [??]).



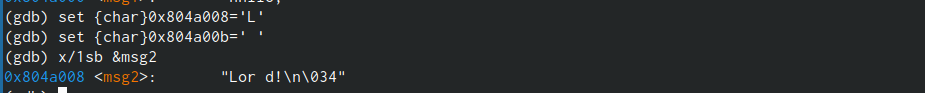
Вывод значений переменных

Изменила значение переменной msg1 с помощью команды set,задав ей в качестве аргумента адрес переменной и имя переменной (рис. [??]).



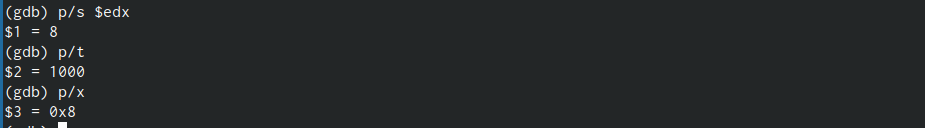
Изменение значения переменной

Изменила значение переменной msg2 с помощью команды set,задав ей в качестве аргумента адрес переменной (рис. [??]).



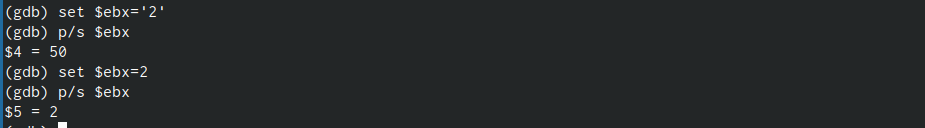
Изменение значения переменной

Вывела в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx (рис. [??]).



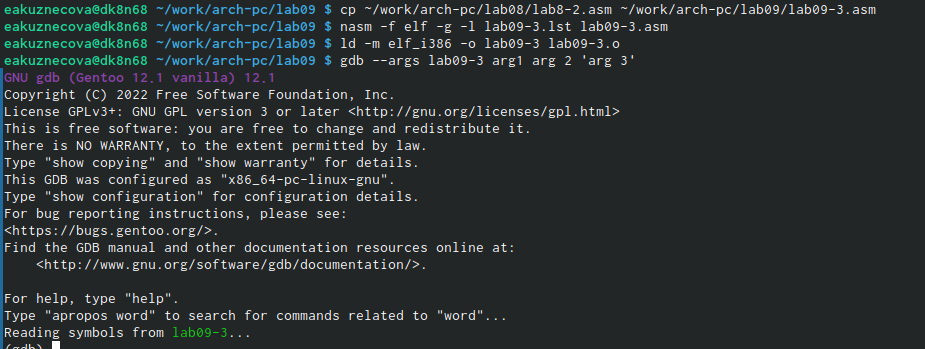
Вывод в различных форматах значение регистра edx

С помощью команды set изменила значение регистра ebx (рис. [??]). При попытке задать строчное значение происходит ошибка. Завершила работу в gdb командами continue,она закончила выполнение программы, и exit, она завершила сеанс gdb.



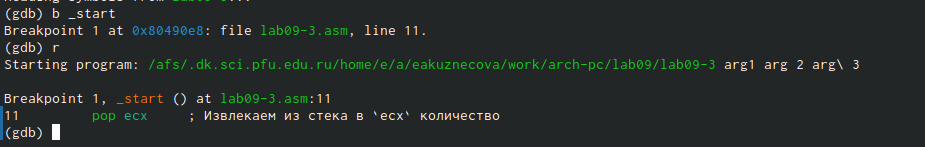
Изменение значения регистра ebx

Скопировала файл из лабораторной 8, создала новый исполняемый файл программы. Для работы с GDB в исполняемый файл добавила отладочную информацию, провела трансляцию программ с ключом ‘-g’. Загрузила исполняемый файл с аргументами в отладчик gdb, использовала ключ –args (рис. [??]).



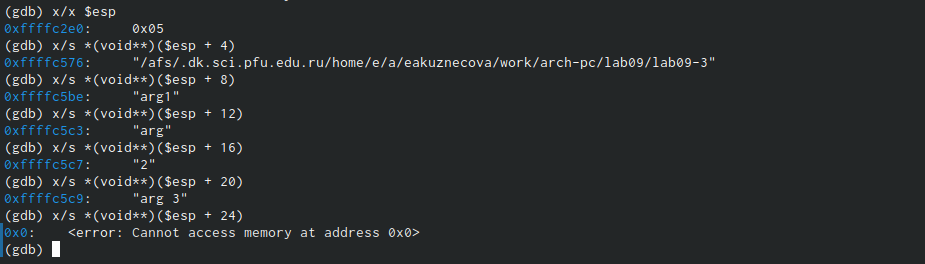
Загрузка исполняемого файла в gdb

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



Установка точки останова

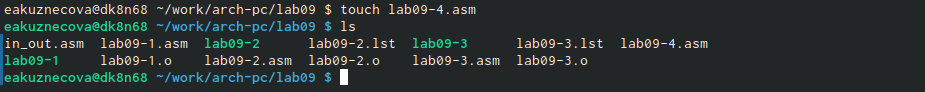
Посмотрела на содержимое того, что расположено по адресу, находящемуся в регистре esp. Увидела, что число аргументов 5 (рис. [??]). Далее посмотрела на все остальные аргументы в стеке. Их адреса располагаются в 4 байтах друг от друга (именно столько занимает элемент стека).



Просмотр в регистре esp

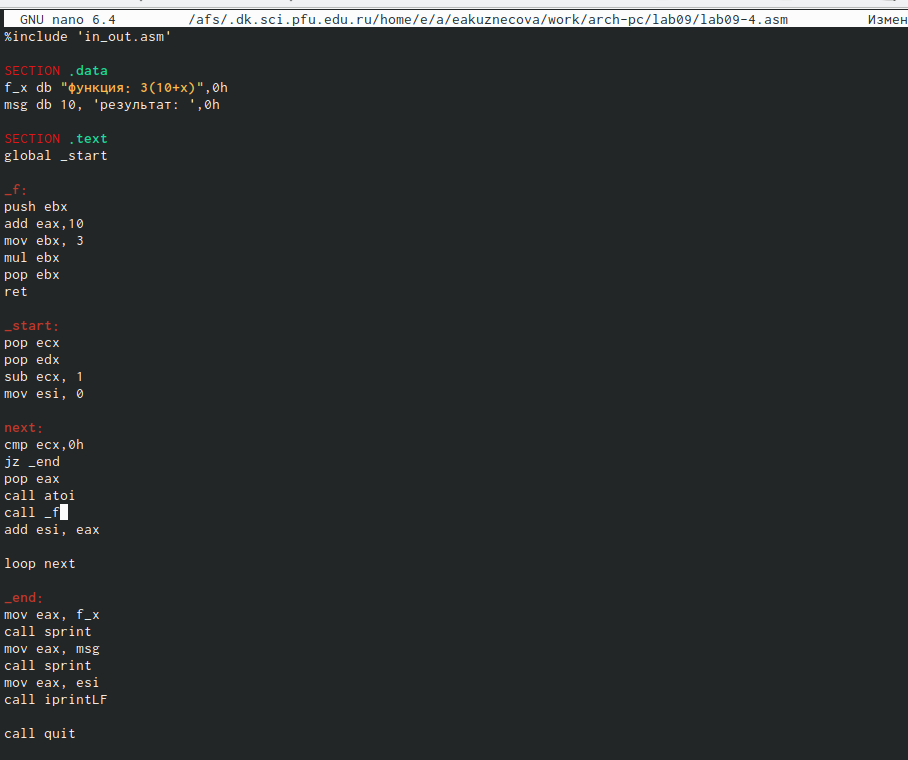
# 5 Выполнение заданий для самостоятельной работы

С помощью утилиты touch создала файл lab09-4.asm (рис. [??]).



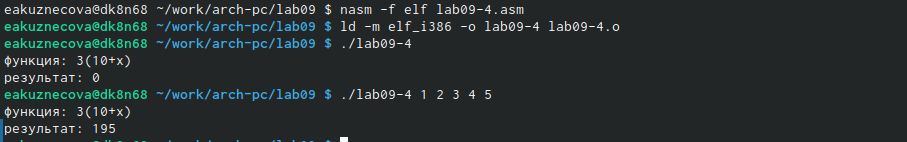
Создание файла

Открыла созданный файл lab09-4.asm, вставила в него программу из 8 лабораторной, но добавила подрограмму для вычисления функции 20 варианта (рис. [??]).



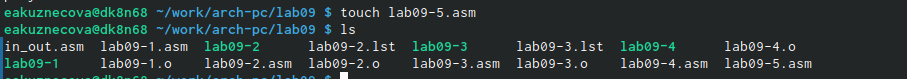
Редактирование файла

Создала новый исполняемый файл программы и запустила его. Результат данной программы являлся правильным (рис. [??]).



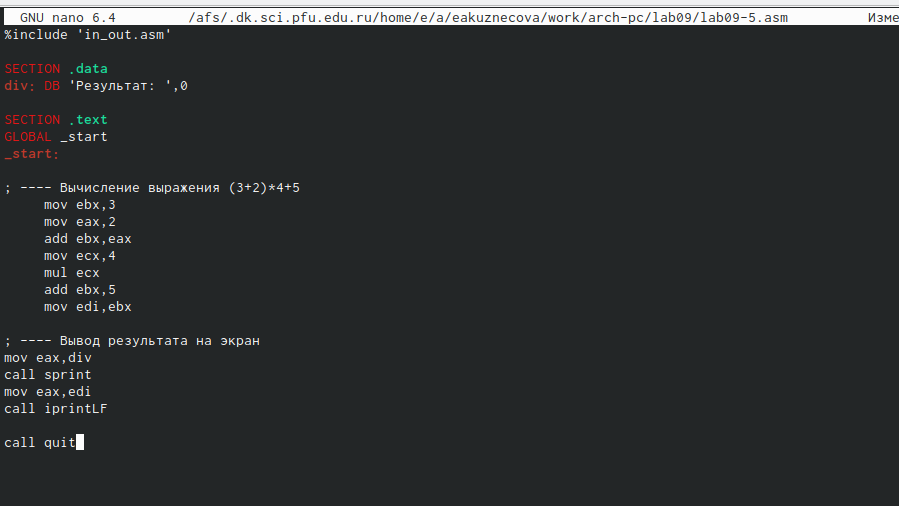
Запуск исполняемого файла

С помощью утилиты touch создала файл lab09-5.asm (рис. [??]).



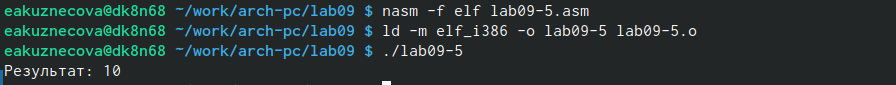
Создание файла

Открыла созданный файл lab09-5.asm, вставила в него программу листинга 9.3 (рис. [??]).



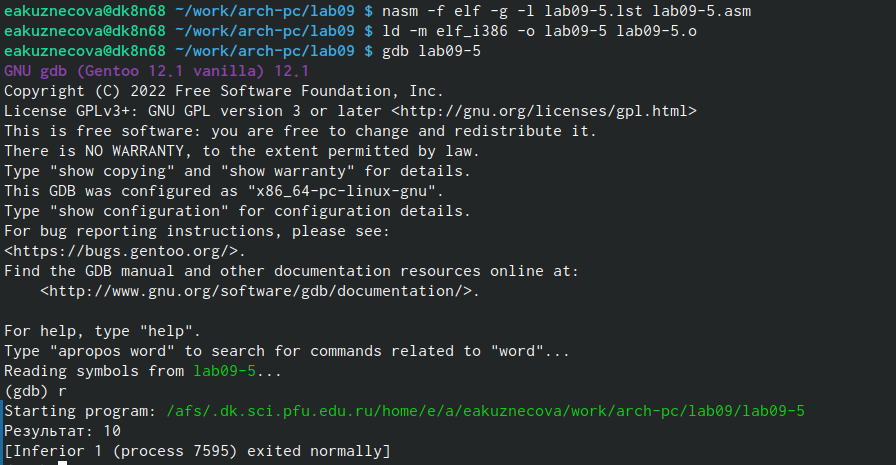
Редактирование файла

Создала новый исполняемый файл программы и запустила его. Результат данной программы являлся неправильным (рис. [??]).



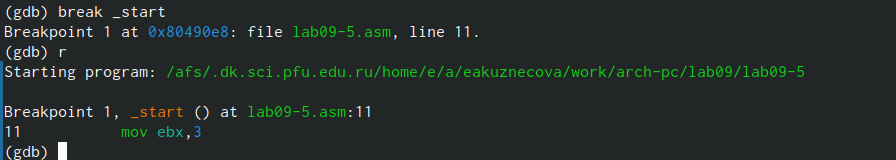
Запуск исполняемого файла

Создала новый исполняемый файл программы. Для работы с GDB в исполняемый файл добавила отладочную информацию, провела трансляцию программ с ключом ‘-g’. Загрузила исполняемый файл в отладчик gdb. Проверила работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r). Программа работает неправильно (рис. [??]).



Загрузка исполняемого файла в gdb и проверка работы программы

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



Запуск программы

Посмотрела дисассимилированный код программы с помощью команды disassemble, начиная с метки \_start (рис. [??]).



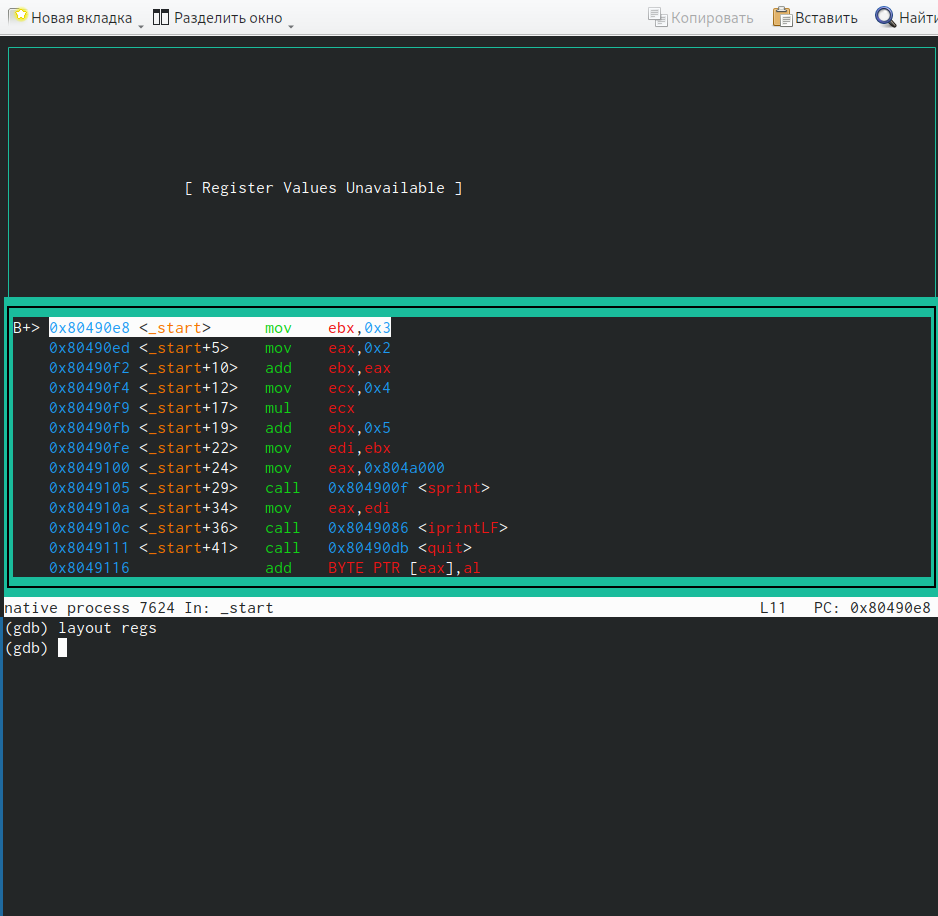
Просмотр дисассимилированного кода программы

Переключилась на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel (рис. [??]).



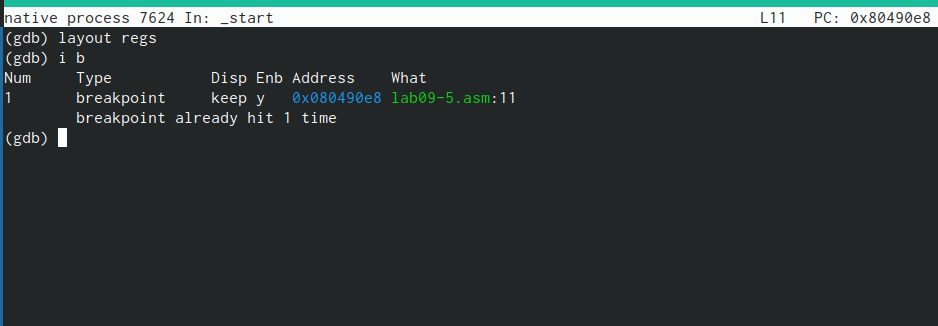
Просмотр дисассимилированного кода программы в режиме intel

Включила режим псевдографики для более удобного анализа программы (рис. [??]).



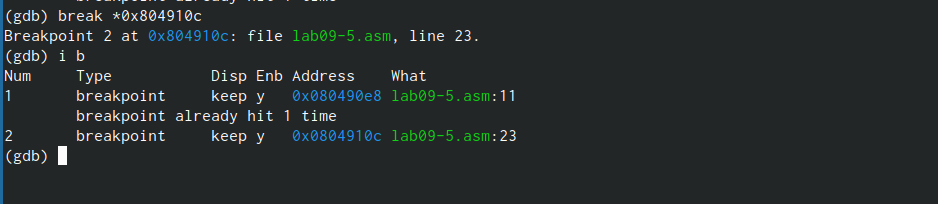
Режим псевдографики

Посмотрела информацию о наших точках останова.Сделала это с помощью команды info breakpoints (кратко i b) (рис. [??]).



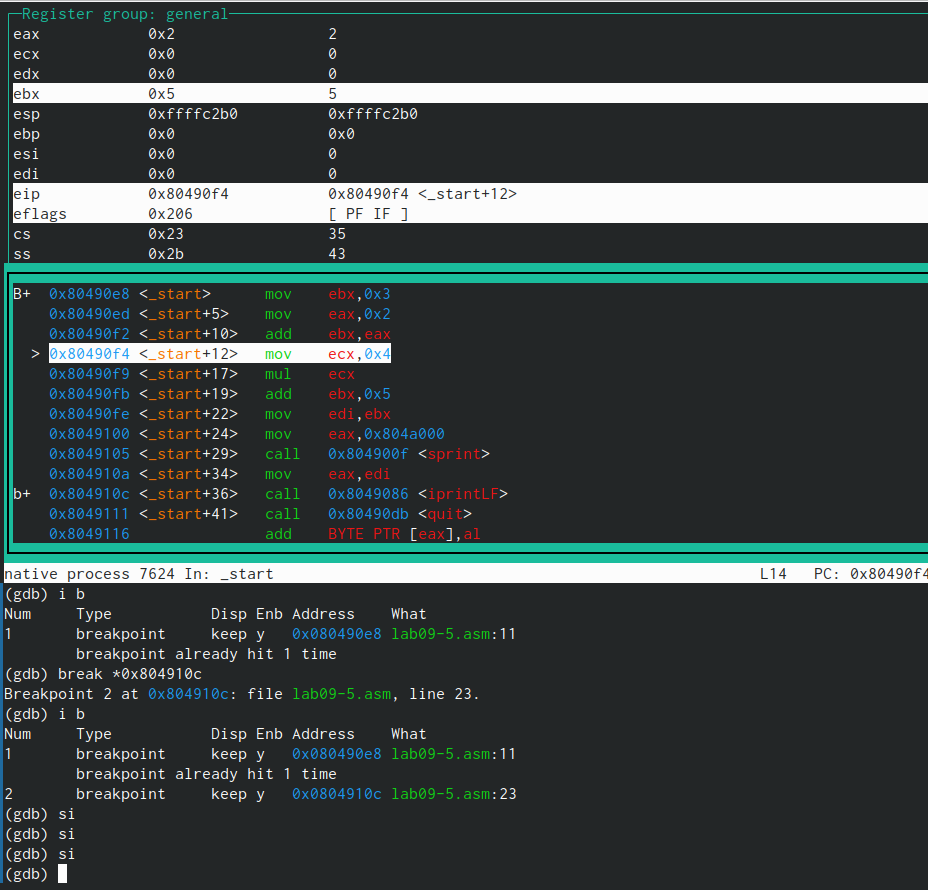
Просмотр точек останова

Добавила еще одну точку останова по адресу инструкции.Посмотрела информацию о всех установленных точках останова (рис. [??]).

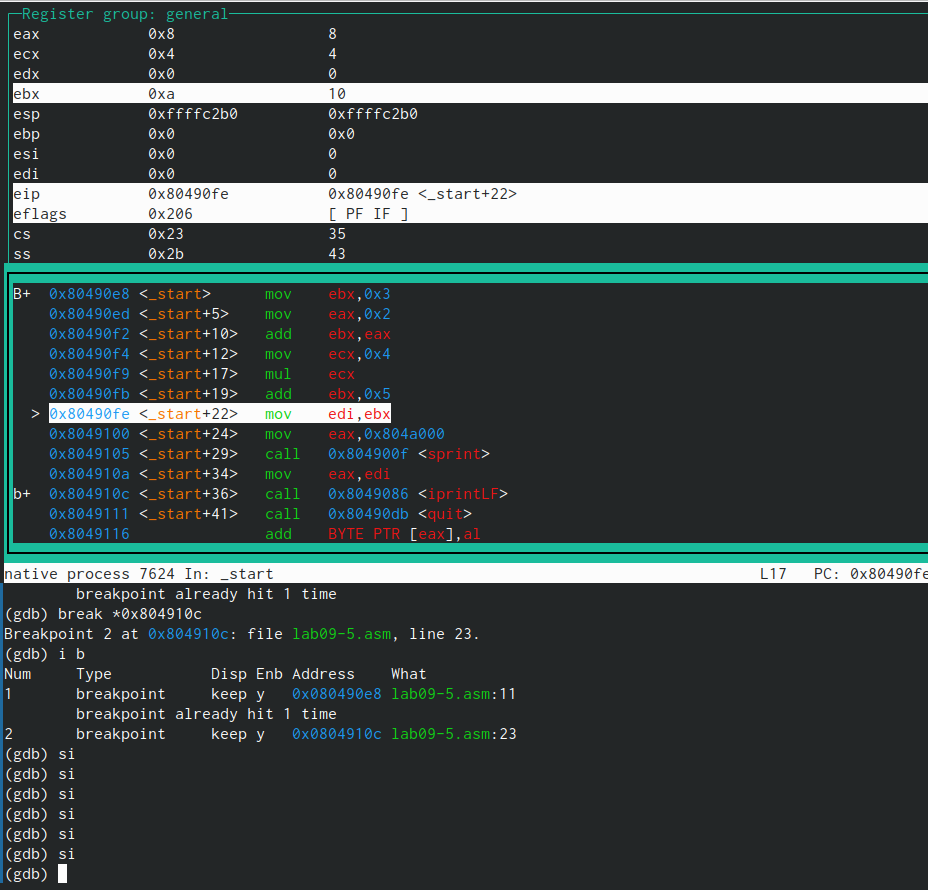


Добавление точки остановы

Выполнила 6 инструкций с помощью команды stepi и нашла все ошибки (рис. [??]) и (рис. [??])



Нахождение ошибок

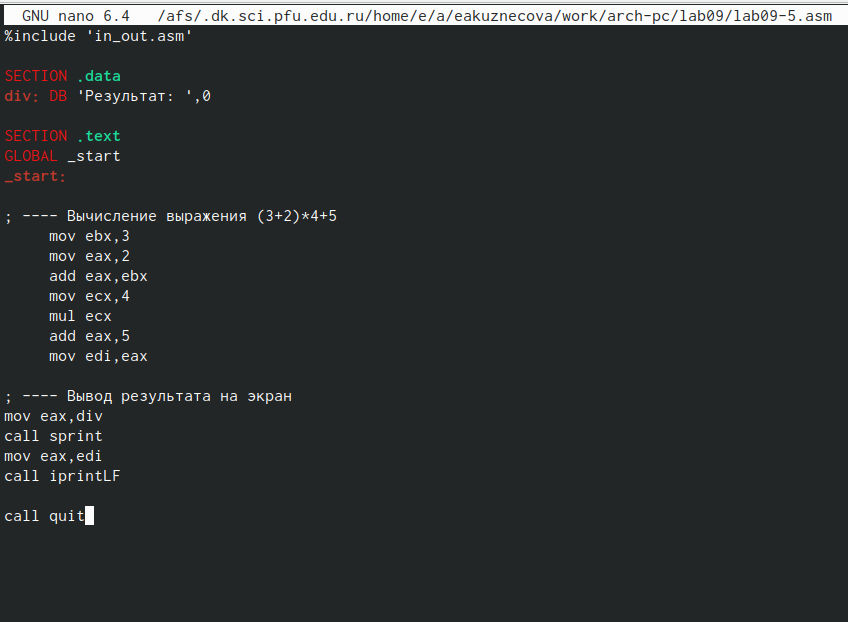


Нахождение ошибок

Ошибка была в строках

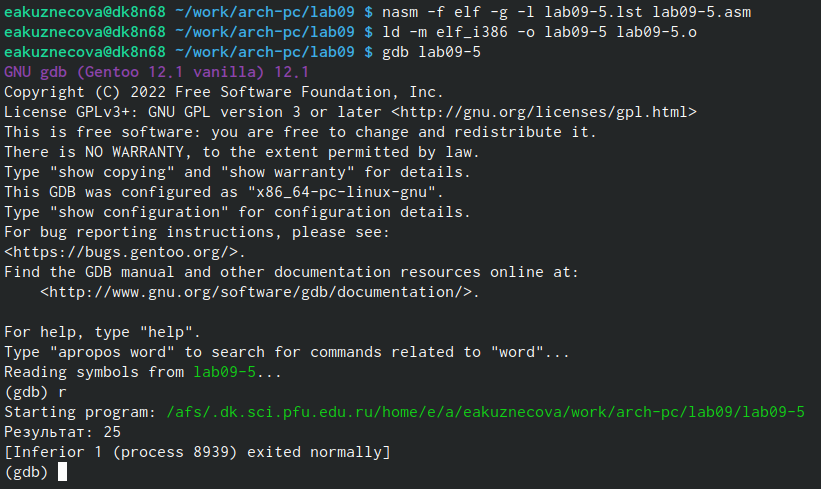
add ebx,eax  
mov ecx,4  
mul ecx  
add ebx,5  
mov edi,ebx

Изменила текст программы lab09-5.asm, чтобы выводился правильный ответ (рис. [??]).



Редактирование файла

Создала новый исполняемый файл программы.Для работы с GDB в исполняемый файл добавила отладочную информацию, провела трансляцию программ с ключом ‘-g’. Загрузила исполняемый файл в отладчик gdb. Проверила работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r). Результат вывелся правильно (рис. [??]).



Загрузка исполняемого файла в gdb и проверка работы программы

# 6 Выводы

В результате выполнения работы, я научилась организовывать код в подпрограммы и познакомилась с базовыми функциями откладчика gdb.