Отчет по лабораторной работе №9

Дисциплина: архитектура компьютера

Адмиральская Александра Андреевна

Содержание

# 1 Цель работы

Целью работы является приобретение навыков написания программ с использованием подпрограмм, а также 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.Добавление точек останова 4.Работа с данными программы в GDB 5.Обработка аргументов командной строки в GDB 6.Задание для самостоятельной работы

# 3 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы.

Например, в табл. 1 приведено краткое описание стандартных каталогов Unix.

Таблица 1: Описание некоторых каталогов файловой системы GNU Linux

| Имя каталога | Описание каталога |
| --- | --- |
| / | Корневая директория, содержащая всю файловую |
| /bin | Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям |
| /etc | Общесистемные конфигурационные файлы и файлы конфигурации установленных программ |
| /home | Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя |
| /media | Точки монтирования для сменных носителей |
| /root | Домашняя директория пользователя root |
| /tmp | Временные файлы |
| /usr | Вторичная иерархия для данных пользователя |

Более подробно про Unix см. в [1–4].

# 4 Выполнение лабораторной работы

Сначала я создаю каталог lab09 и файл lab09-1.asm (рис. 1).



Рис. 1: Создание каталога и файла

Открываю созданный файл и ввожу в него тест программы из листинга 9.1 (рис. 2).

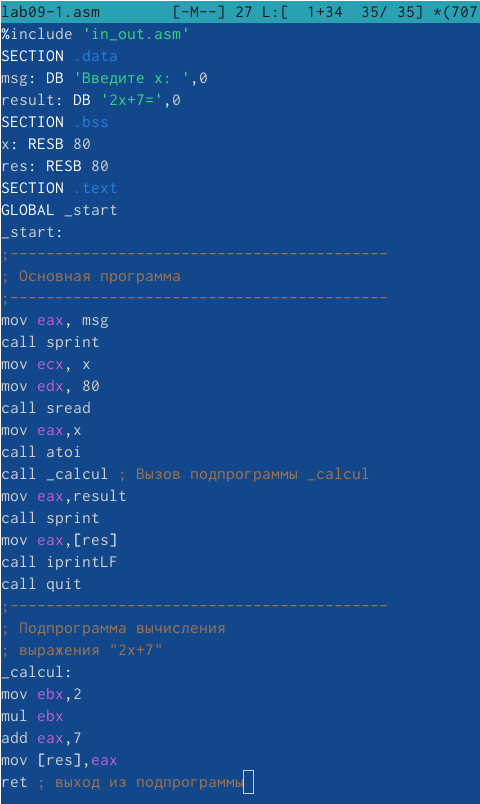


Рис. 2: Текст программы в файле

Создаю исполняемый файл и запусаю его (рис. 3).

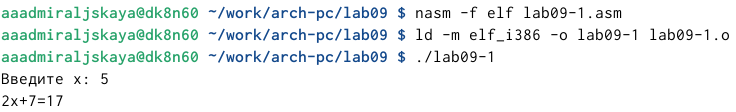


Рис. 3: Запуск файла

Изменяю текст программы так, чтобы она вычисляла значение выражения f(g(x))(рис. 4).

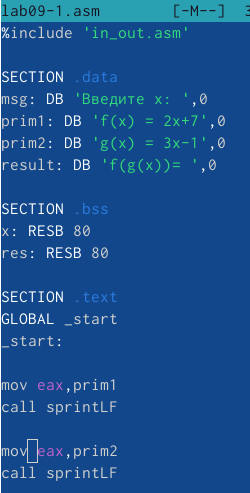


Рис. 4: Измененный текст программы

Создаю исполняемый файл и запусаю его (рис. 5).

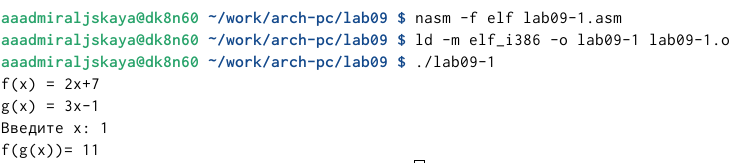


Рис. 5: Запуск файла

## 4.1 Отладка программам с помощью GDB

Создаю файл lab9-2.asm (рис. 6).

Рис. 6: Создание файла

Рис. 6: Создание файла

Ввожу в него тескт программы из листинга 9.2 (рис. 7).

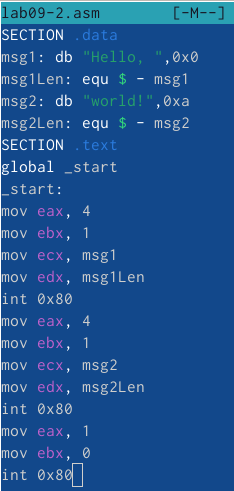


Рис. 7: Текст второй программы

Далее получаю исполнямый файл с помощью ключа -g и загружаю этот файл в отладчик gdb (рис. 8).

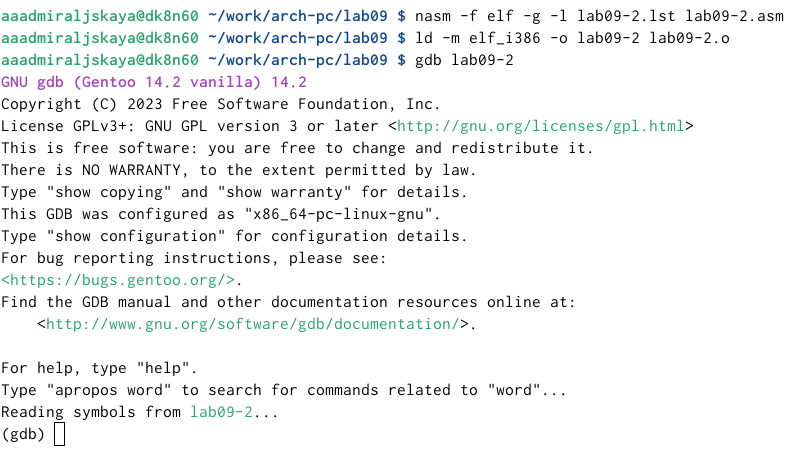


Рис. 8: Загрузка файла в откладчик

Проверяю работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r) (рис. 9).

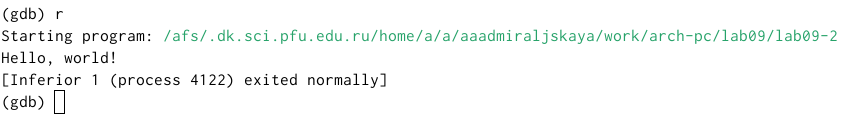


Рис. 9: Проверка работы программы

Теперь устанавливаю брейкпоинт на метку \_start и запускаю программу (рис. 10).



Рис. 10: Установка брейкпоинта на метку \_start и запуск программы

Посмотриваю дисассимилированный код программы с помощью команды disassemble, начиная с метки \_start (рис. 11).

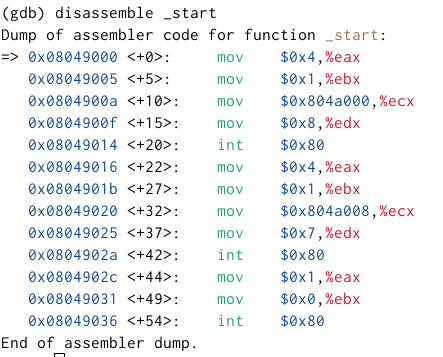


Рис. 11: Дисассимплированный код

Переключаюсь на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel (рис. 12).

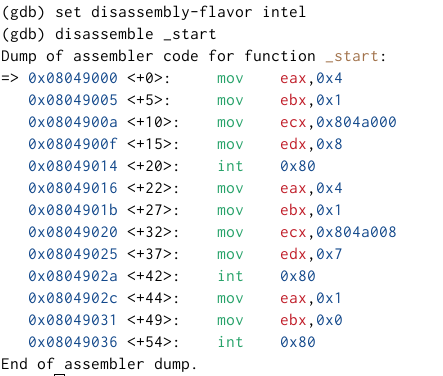


Рис. 12: Intel’овское отображение

Для удобства включаю режим псевдографики (рис. 13).

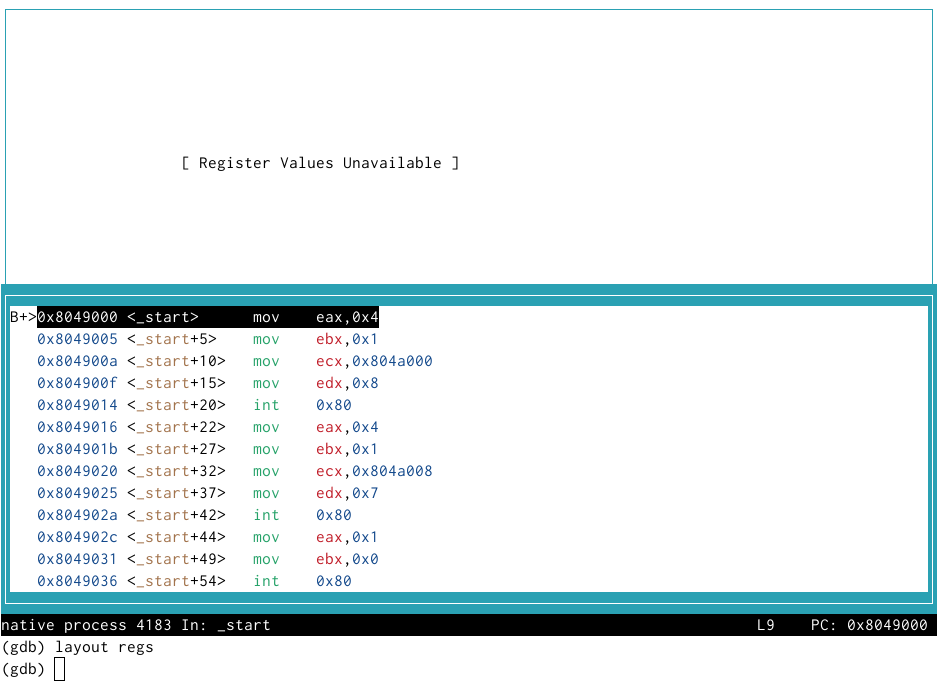


Рис. 13: Псевдографика

## 4.2 Добавление точек останова

На предыдущих шагах была установлена точка остановки по имени метки (\_start),проверяю это с помощью команды info breakpoints (кратко i b). Также устанавливаю еще одну точку остановки по адресу инструкции и снова смотрю информацию о всех установленных метках(рис. 14).

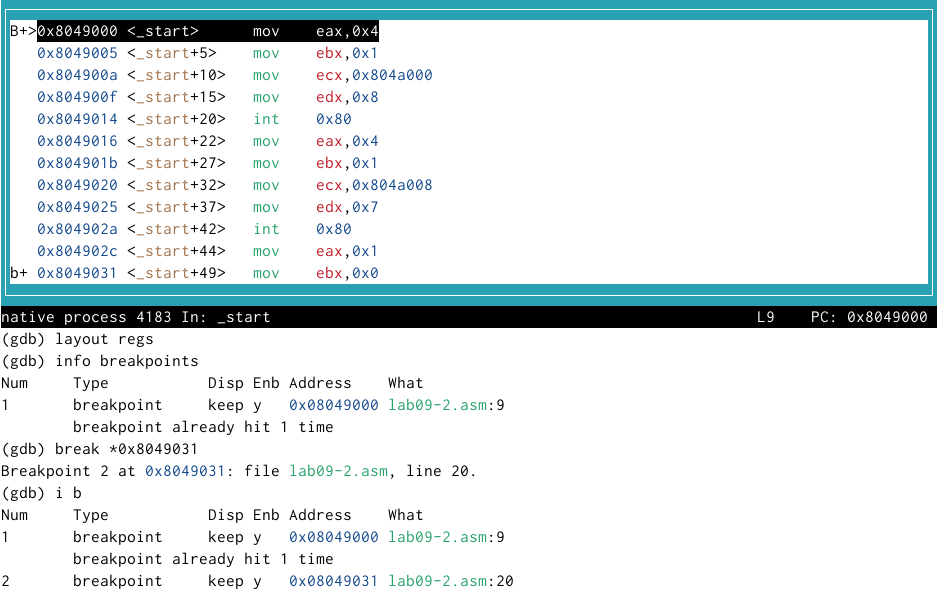


Рис. 14: Проверка меток

## 4.3 Работа с данными программы в GDB

С помощью команды si я посмотрела регистры и изменила их (рис. 15).

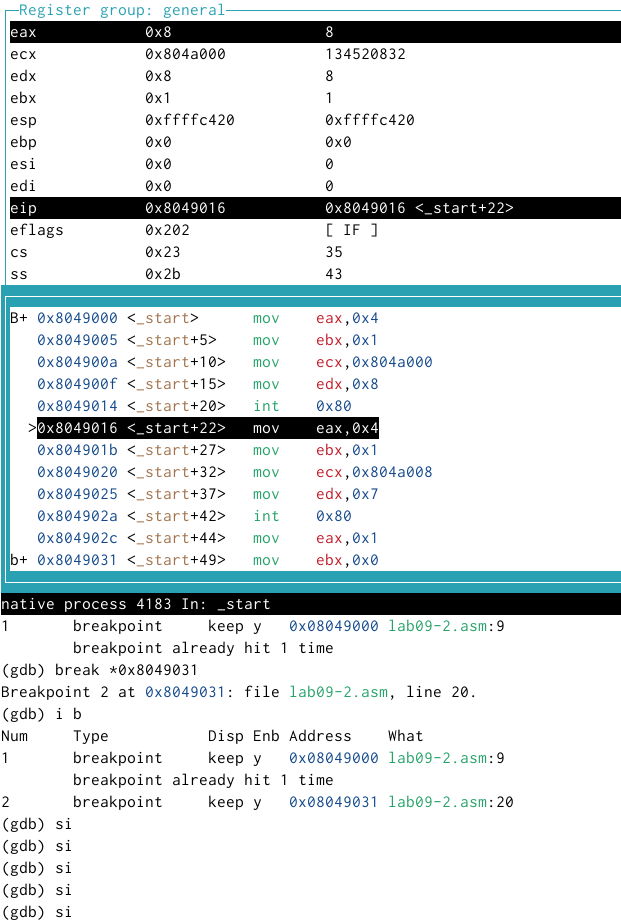


Рис. 15: Измение регистров

Далее я посмотрела значения переменных msg1 и msg2 (рис. 16).

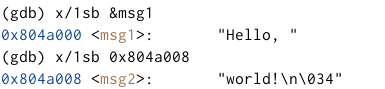


Рис. 16: Просмотр значений переменных

С помощью команды set я изменяю значения переменных msg1 и msg2 (рис. 17).

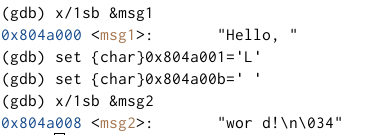


Рис. 17: Изменение значений переменных

Теперь вывожу значение регистров ecx и eax с помощью (рис. 18).

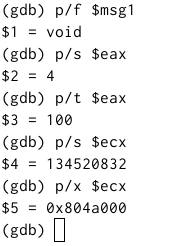


Рис. 18: Значение регистров ecx и eax

Теперь изменяю значение регистра ebx (рис. 19). Команда выводит два разных значения, так как в первый раз мы вносим значение 2, а во второй раз регистр равен двум.

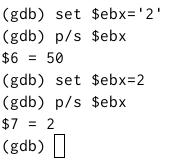


Рис. 19: Значение регистров ebx

Завершаю выполнение программы с помощью команды continue (сокращенно c) и выхожу из GDB с помощью команды quit (сокращенно q).

## 4.4 Обработка аргументов командной строки в GDB

Копирую файл lab8-2.asm и переименовываю его (рис. 20).

Рис. 20: Копирование файла

Рис. 20: Копирование файла

Создаю исполняемый файл (рис. 21).

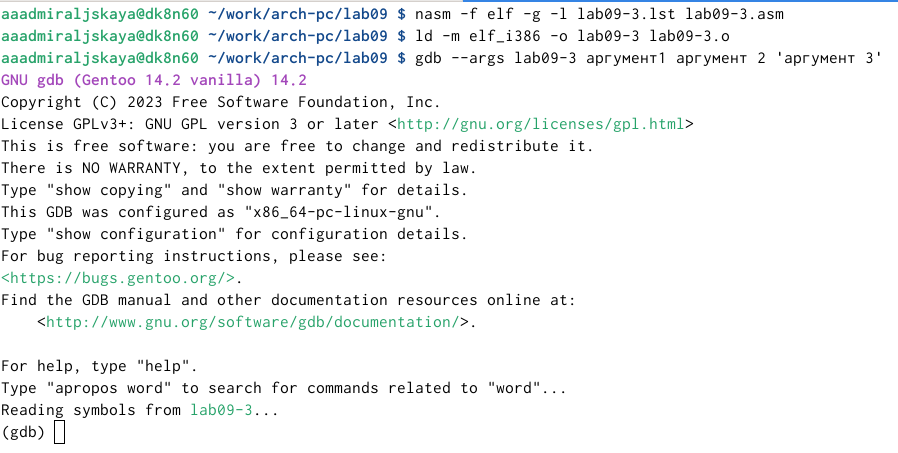


Рис. 21: Создание исполняемого файла

Запускаю файл, указав аргументы (рис. 22).

Рис. 22: Запуск файла

Рис. 22: Запуск файла

Ставлю метку на \_start и запустила файл (рис. 23).

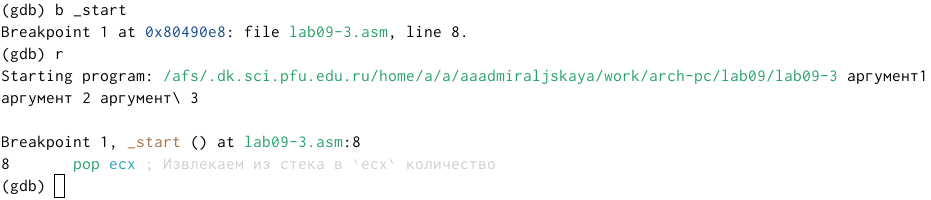


Рис. 23: Запуск файла lab10-3 через метку

Проверяю адрес вершины стека, там хранится 5 элементов (рис. 24).

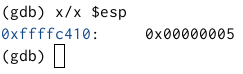


Рис. 24: Адрес вершины стека

Теперь просматриваю все позиции стека. По первому адресу хранится адрес, а в остальных адресах хранятся элементы. при этом 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. 25).

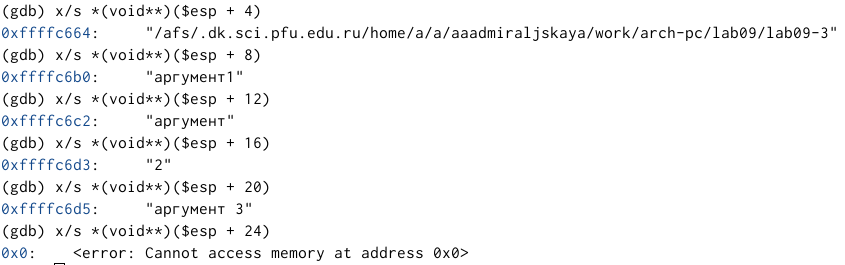


Рис. 25: Все позиции стека

## 4.5 Задание для самостоятельной работы

1. Копирую файл lab8-4.asm из лаборатоной работы №8 в папку для лаборатоной №9 с названием lab9-4.asm и изменяю текст программы так, чтобы она вычисляла значение функции как подпрограмму (рис. 26).

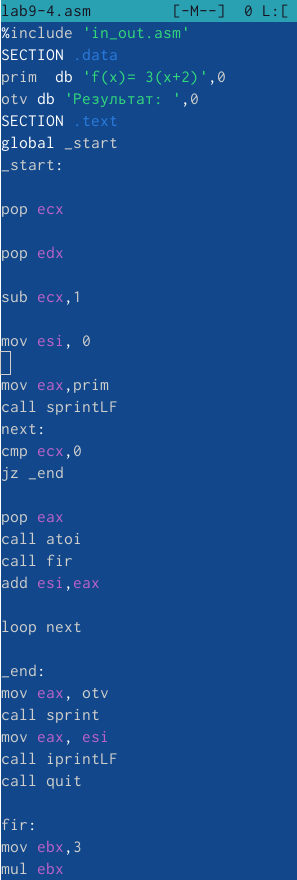


Рис. 26: Текст программы

Создаю исполняемый файл и запускаю его (рис. 27).

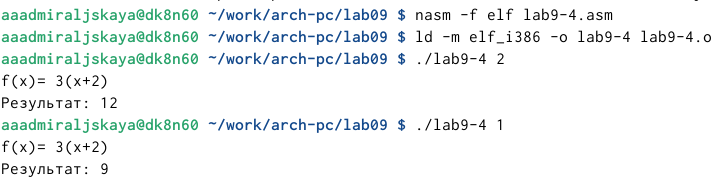


Рис. 27: Запуск файла

1. Создаю файл lab9-5.asm (рис. 28).

|  |
| --- |
| Рис. 28: Создание файла |

Рис. 28: Создание файла

Листинг для файла lab09-4:

%include ‘in\_out.asm’ SECTION .data prim db ‘f(x)= 3(x+2)’,0 otv db ‘Результат:’,0 SECTION .text global \_start \_start:

pop ecx

pop edx

sub ecx,1

mov esi, 0

mov eax,prim call sprintLF next: cmp ecx,0 jz \_end

pop eax call atoi call fir add esi,eax

loop next

\_end: mov eax, otv call sprint mov eax, esi call iprintLF call quit

fir: mov ebx,3 mul ebx add eax,6 ret

Открываю созданный файл и ввожу в него текст программы из листинга 9.3 (рис. 29).

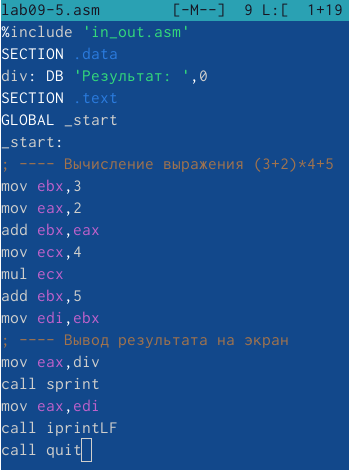


Рис. 29: Текст программы в файле

Создаю исполняемый файл и запускаю его (рис. 30). Ошибка арифметическая, так как вместо 25,программа выводит 10.

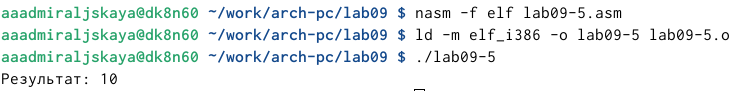


Рис. 30: Запуск программы

После появления ошибки, я запускаю программу в отладчике (рис. 31).

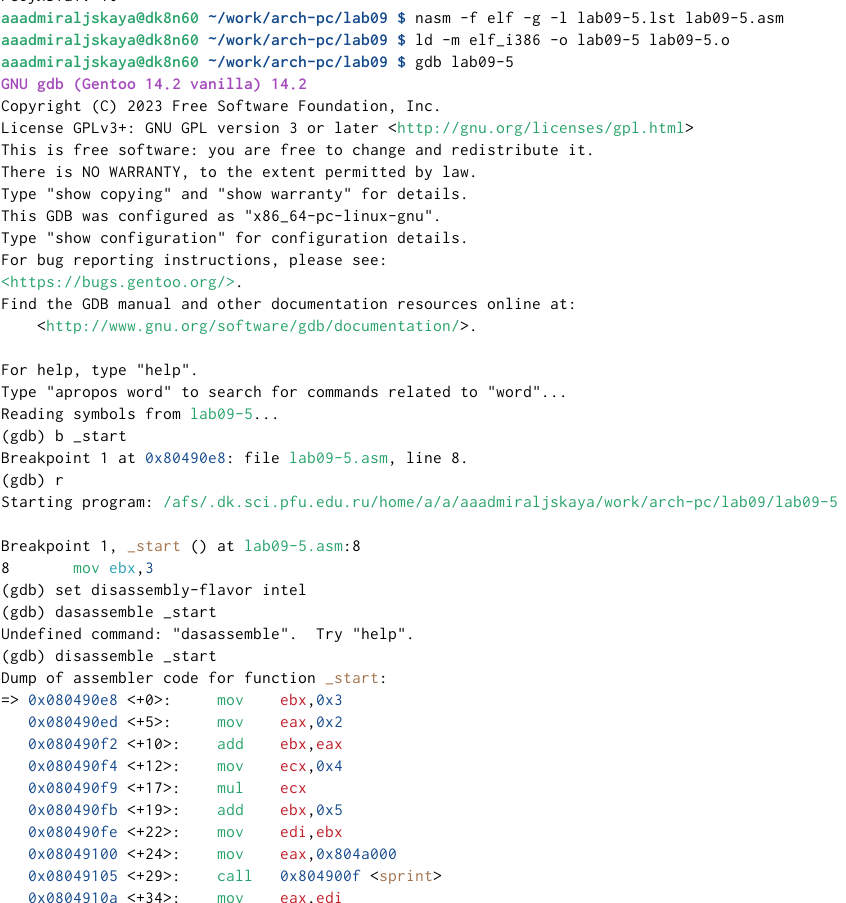


Рис. 31: Запуск программы в отладчике

Я открыла регистры, поняла что регистры стоят не на своих местах и исправила это (рис. 32).

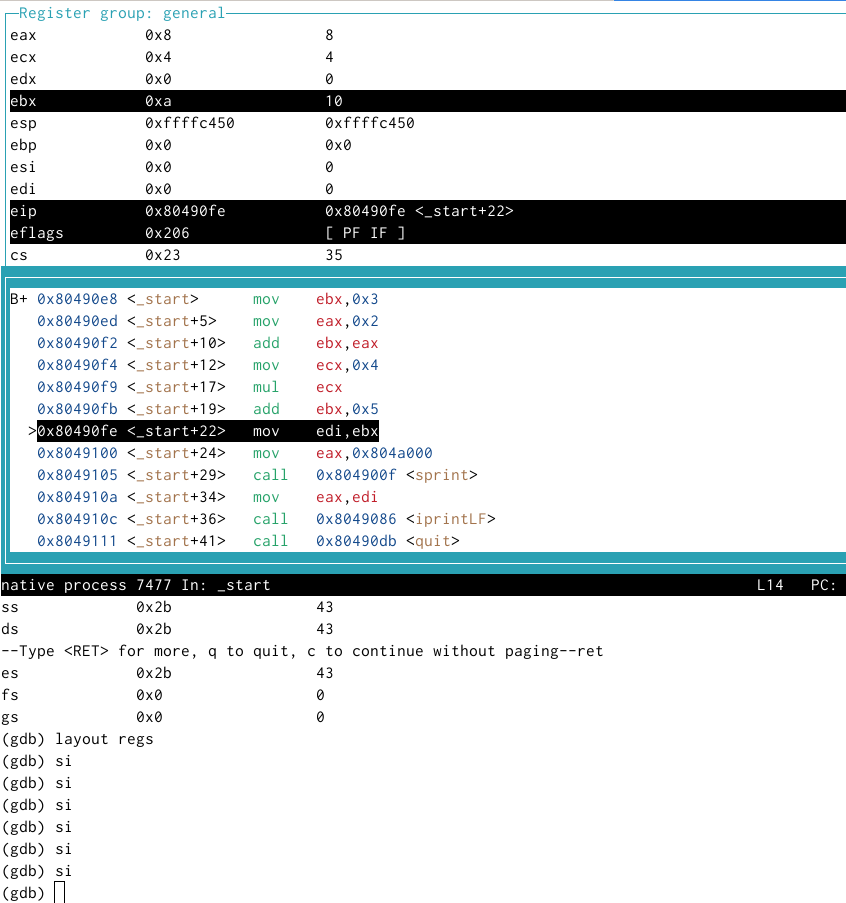


Рис. 32: Анализ регистров

Я изменила регистры(рис. 33) и запустила программу. Программа вывела 25, то есть все работает правильно.

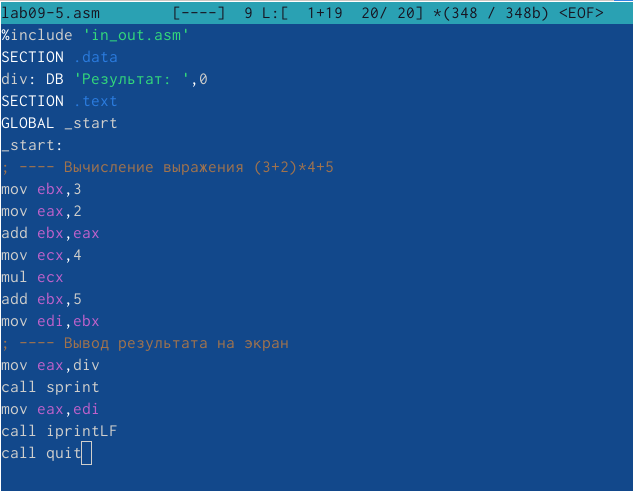


Рис. 33: Изменение программы

Листинг файла lab09-5:

%include ‘in\_out.asm’ SECTION .data div: DB ‘Результат:’,0 SECTION .text GLOBAL \_start \_start: ; —- Вычисление выражения (3+2)\*4+5 mov ebx,3 mov eax,2 add ebx,eax mov ecx,4 mul ecx add ebx,5 mov edi,ebx ; —- Вывод результата на экран mov eax,div call sprint mov eax,edi call iprintLF call quit

# 5 Выводы

При выполнении данной лабораторной работы я приобрела навыки написания программ с использованием подпрограмм, а также ознакомилась с методами отладки при помощи GDB и его основными возможностями.

# Список литературы

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.

2. Robbins A. Bash Pocket Reference. O’Reilly Media, 2016. 156 с.

3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.

4. Newham C. [Learning the bash Shell: Unix Shell Programming](http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658). O’Reilly Media, 2005. 354 с.