Отчёт по лабораторной работе №9

Дисциплина: архитектура компьютера

Учаева Алёна Сергеевна

Содержание

Список иллюстраций

# 1 Цель работы

Приобрести навыки написания программ с использованием подпрограмм. Ознакомиться с методами отладки при помощи GDB и его основными возможностями.

# 2 Задание

1. Реализация подпрограмм в NASM
2. Отладка программам с помощью GDB
3. Задание для самостоятельной работы

# 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. 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.

# 4 Выполнение лабораторной работы

## 4.1 Реализация подпрограмм в NASM

Создаю каталог для выполнения лабораторной работы № 9, перехожу в него и со- здайте файл lab09-1.asm (рис. 1).

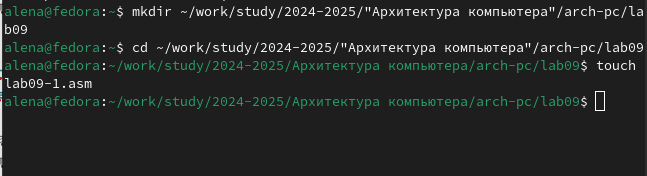


Рис. 1: Создание файла

Ввожу в файл lab09-1.asm текст программы из листинга 9.1. Создаю исполняемый файл и проверяю его работу. Программа выполняет вычисление функции (рис. 2).

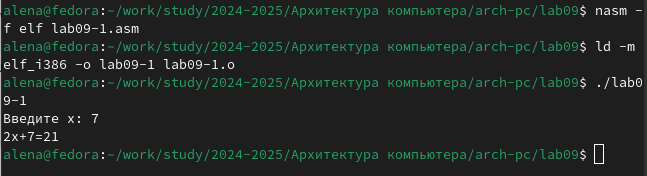


Рис. 2: Запуск исполняемого файла

Изменяю текст программы, добавив подпрограмму \_subcalcul в подпрограмму \_calcul. Программа вычисляет значение функции для выражения f(g(x)) (рис. 3).

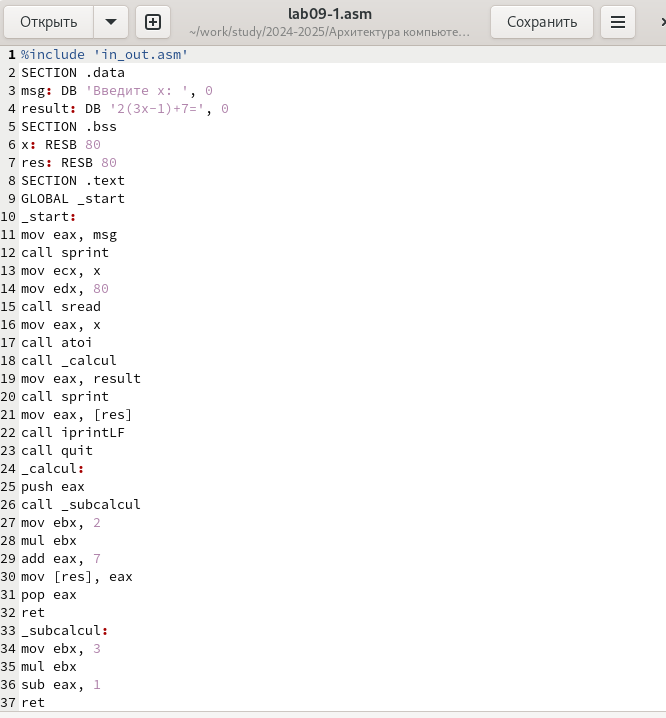


Рис. 3: Редактирование файла

Создаю исполняемый файл и проверяю его работу. Программа выполняет вычисление функции (рис. 4).

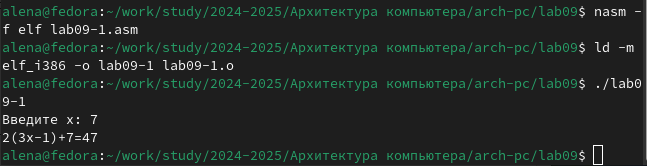


Рис. 4: Запуск исполняемого файла

Код программы:

%include 'in\_out.asm'  
SECTION .data  
msg: DB 'Введите x: ', 0  
result: DB '2(3x-1)+7=', 0  
SECTION .bss  
x: RESB 80  
res: RESB 80  
SECTION .text  
GLOBAL \_start  
\_start:  
mov eax, msg  
call sprint  
mov ecx, x  
mov edx, 80  
call sread  
mov eax, x  
call atoi  
call \_calcul  
mov eax, result  
call sprint  
mov eax, [res]  
call iprintLF  
call quit  
\_calcul:  
push eax  
call \_subcalcul  
mov ebx, 2  
mul ebx  
add eax, 7  
mov [res], eax  
pop eax  
ret  
\_subcalcul:  
mov ebx, 3  
mul ebx  
sub eax, 1  
ret

## 4.2 Отладка программам с помощью GDB

Создаю файл lab09-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!) (рис. 5).

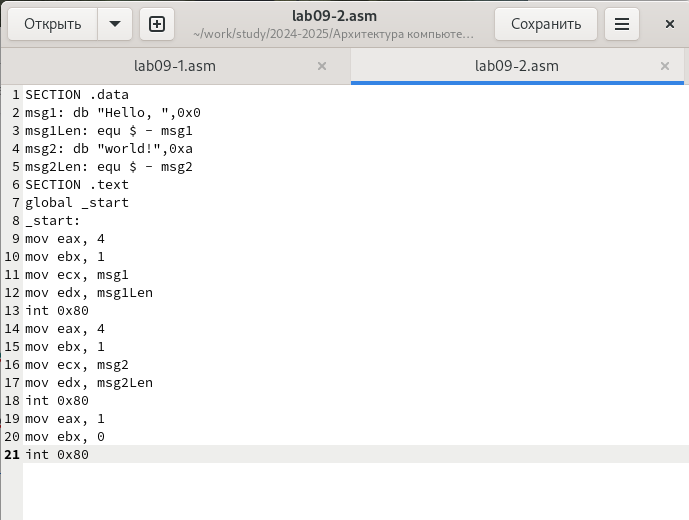


Рис. 5: Создание файла

Транслирую с созданием файла листинга и отладки, компоную и запускаю в отладчике, программа работает корректно (рис. 6).

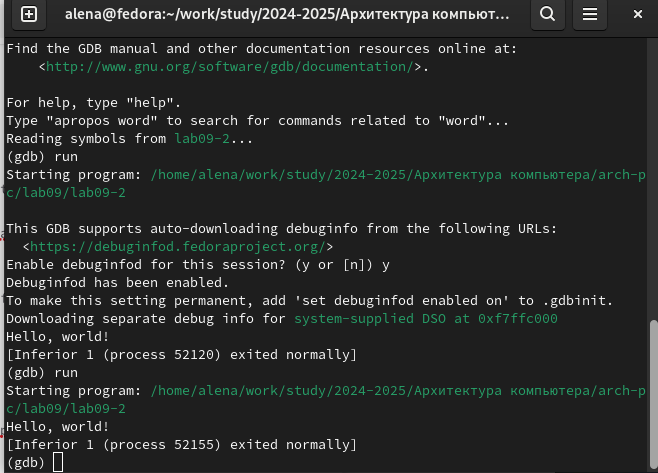


Рис. 6: Проверка программы отладчиком

Для более подробного анализа программы устанавливаю брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запускаю её (рис. 7).

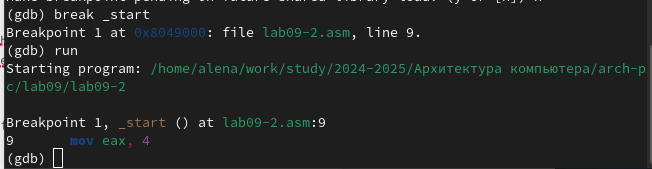


Рис. 7: Запуск отладчика с брейкпоинтом

Далее смотритю дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. 8).

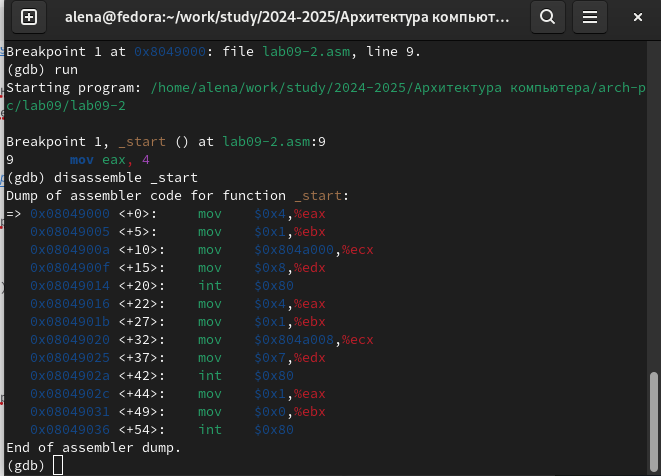


Рис. 8: Дисассимилирование программы

Переключаюсь на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel (рис. 9).



Рис. 9: Дисассимилирование программы

Различия между синтаксисом ATT и Intel заключаются в порядке операндов (ATT - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (ATT - pазмер операндов указывается явно с помощью суффиксов, непосредственные операнды предваряются символом $; Intel - Размер операндов неявно определяется контекстом, как ax, eax, непосредственные операнды пишутся напрямую), именах регистров(ATT - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

Далее включаю режим псевдографики для более удобного анализа программы (рис. 10).

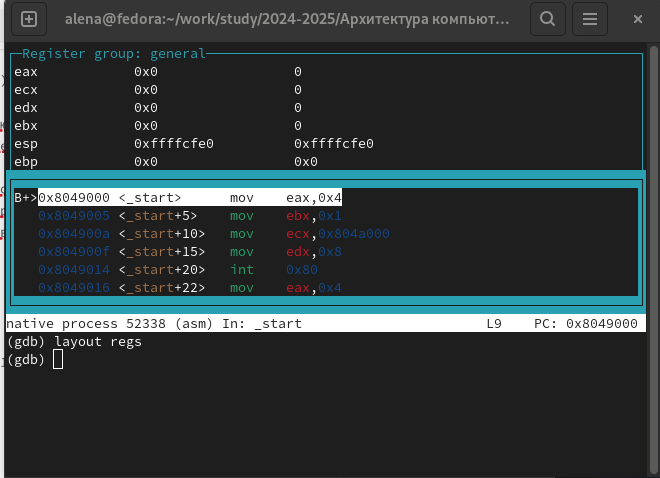


Рис. 10: Режим псевдографики

## 4.3 Добавление точек останова

Проверяю в режиме псевдографики, что брейкпоинт сохранился (рис. 11).

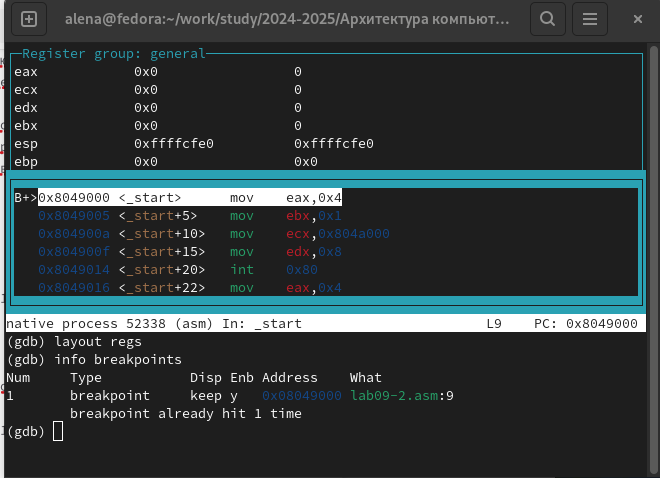


Рис. 11: Режим псевдографики

Устаналиваю еще одну точку останова по адресу инструкции (рис. 12).

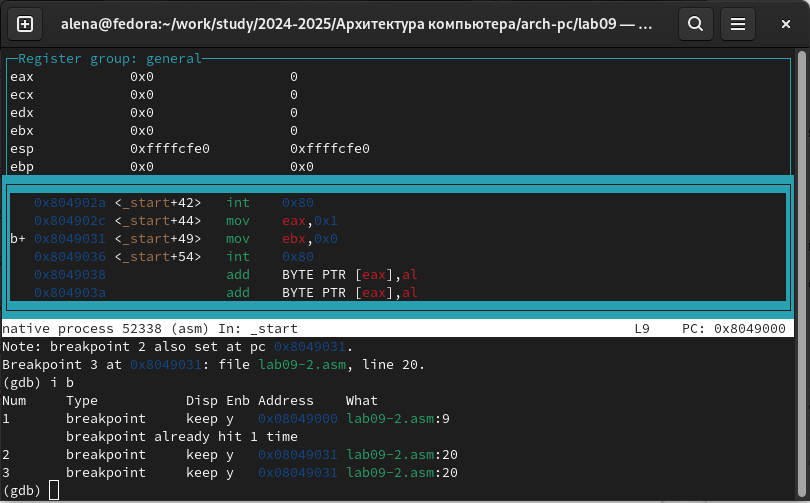


Рис. 12: Установка точки останова

### 4.3.1 Работа с данными программы в GDB

Посмотреть содержимое регистров также можно с помощью команды info registers (рис. 13).

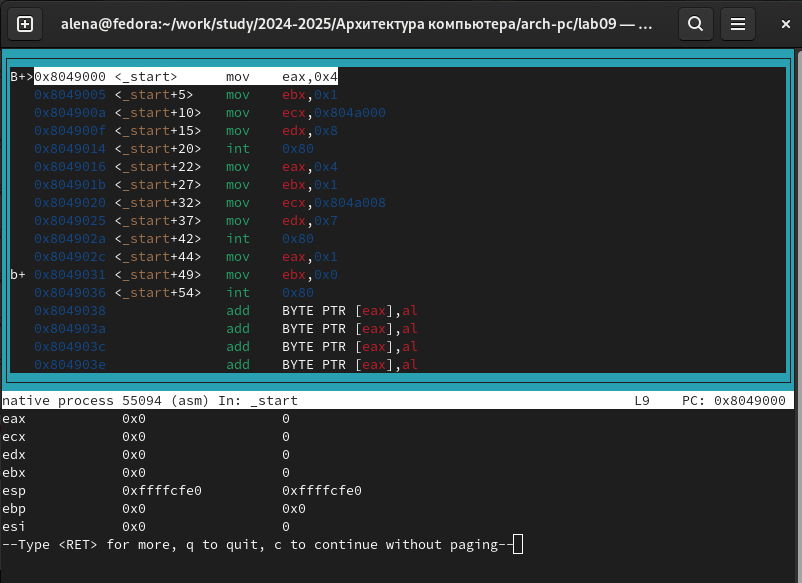


Рис. 13: Просмотр содержимого регистров

Смотрю содержимое переменных по имени и по адресу (рис. 14).

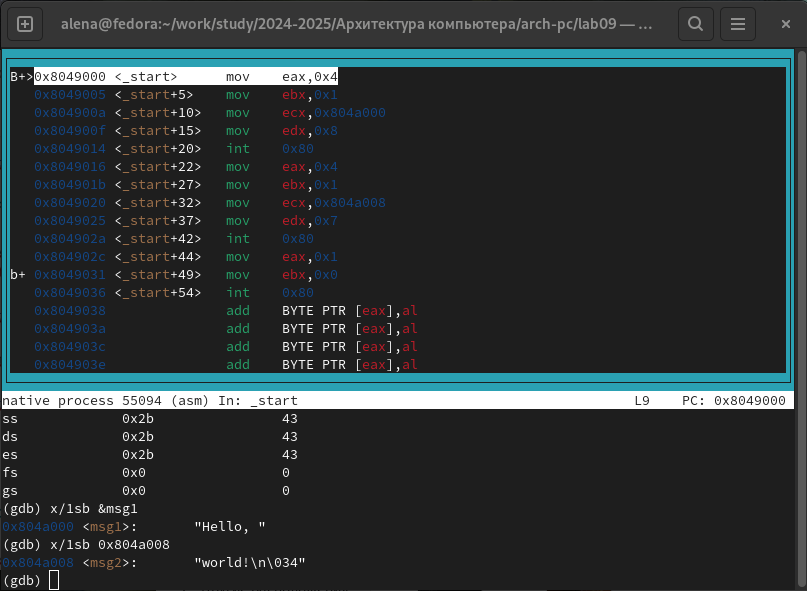


Рис. 14: Просмотр содержимого переменных

Меняю содержимое переменных по имени и по адресу (рис. 15).

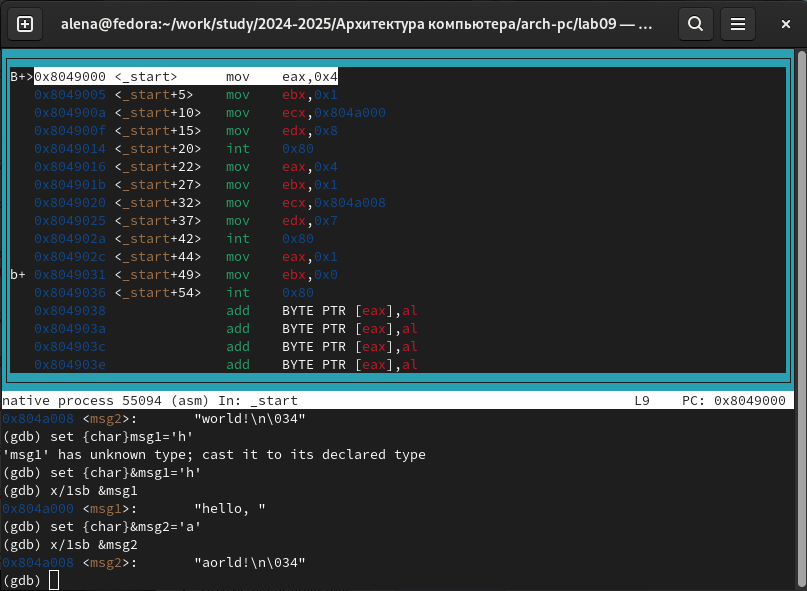


Рис. 15: Изменение содержимого переменных

Вывожу в различных форматах значение регистра edx (рис. 16).

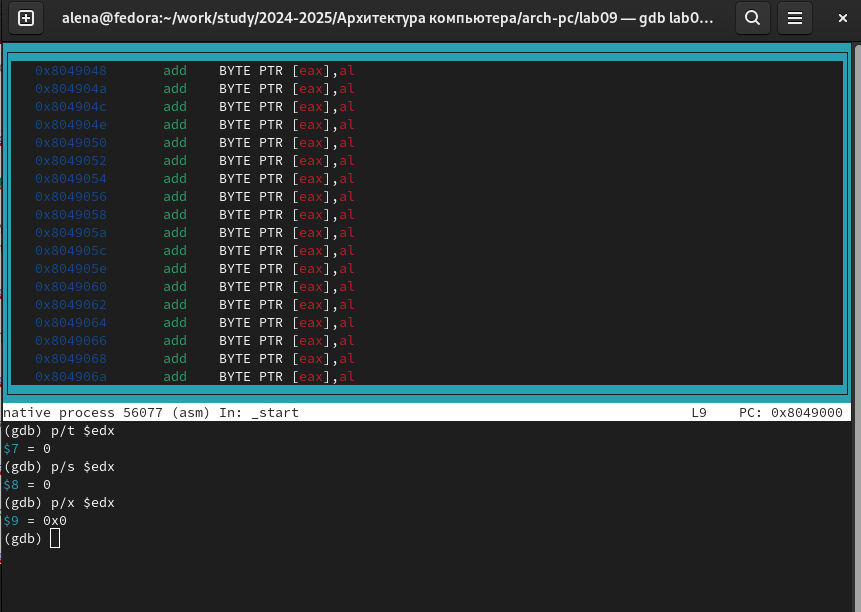


Рис. 16: Просмотр содержимого регистров

С помощью команды set меняю содержимое регистра ebx (рис. 17).

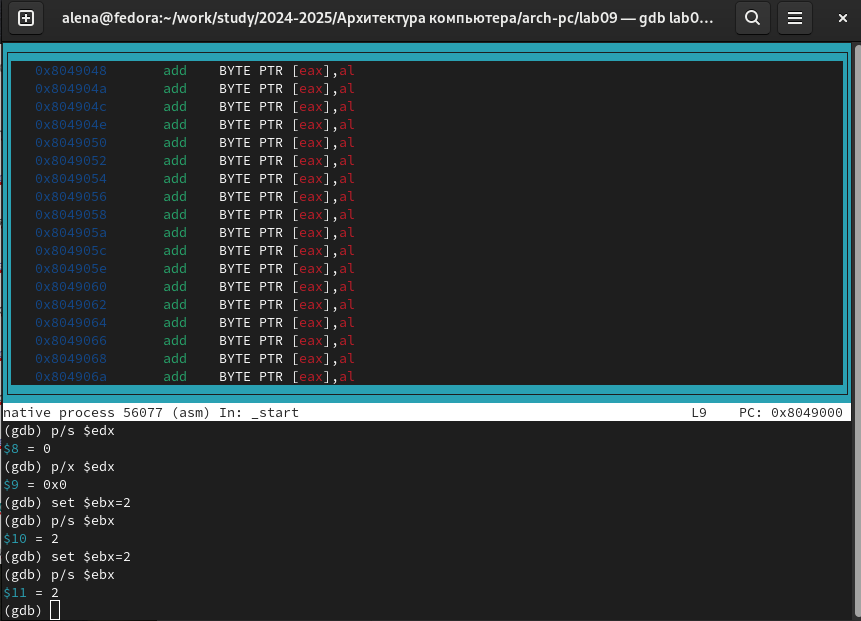


Рис. 17: Редактирование содержимого регистра ebx

### 4.3.2 Обработка аргументов командной строки в GDB

Копирую файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm и создаю исполняемый файл (рис. 18).

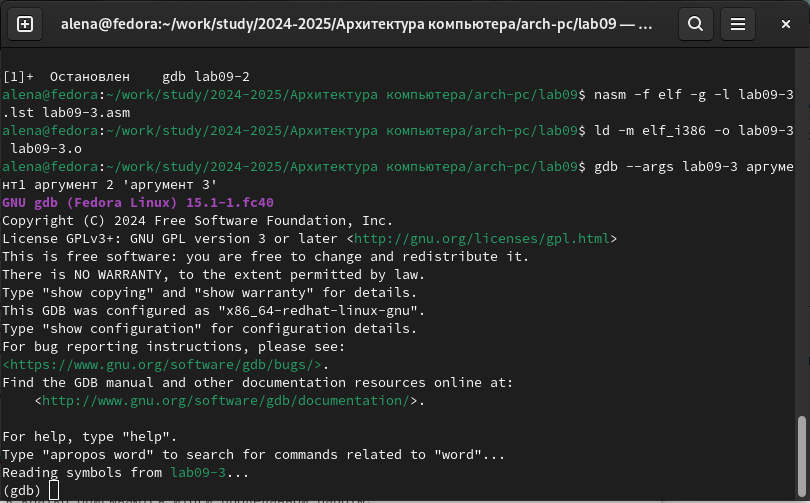


Рис. 18: Создание исполняемого файла

Устанавливаю точку останова перед первой инструкцией в программе и запускаю ее (рис. 19).

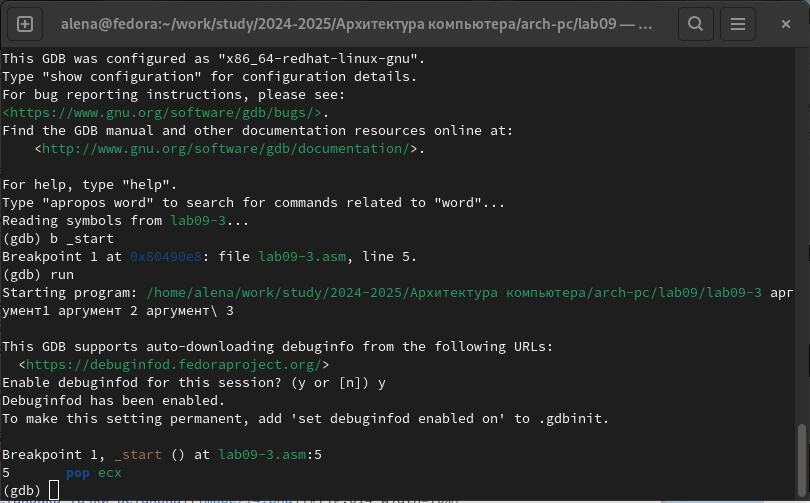


Рис. 19: Установка точки останова

Запускаю программу с режиме отладки с указанием аргументов, указываю брейкпопнт и запускаю отладку. Проверяю работу стека, изменяя аргумент команды просмотра регистра esp на +4, число обусловлено разрядностью системы, а указатель void занимает как раз 4 байта, ошибка при аргументе +24 означает, что аргументы на вход программы закончились (рис. 20).

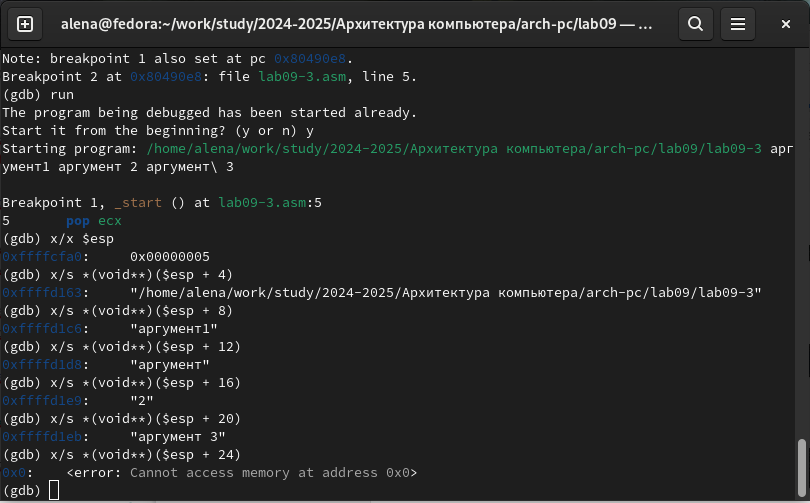


Рис. 20: Запуск отладчика

## 4.4 Задание для самостоятельной работы

Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. 21).

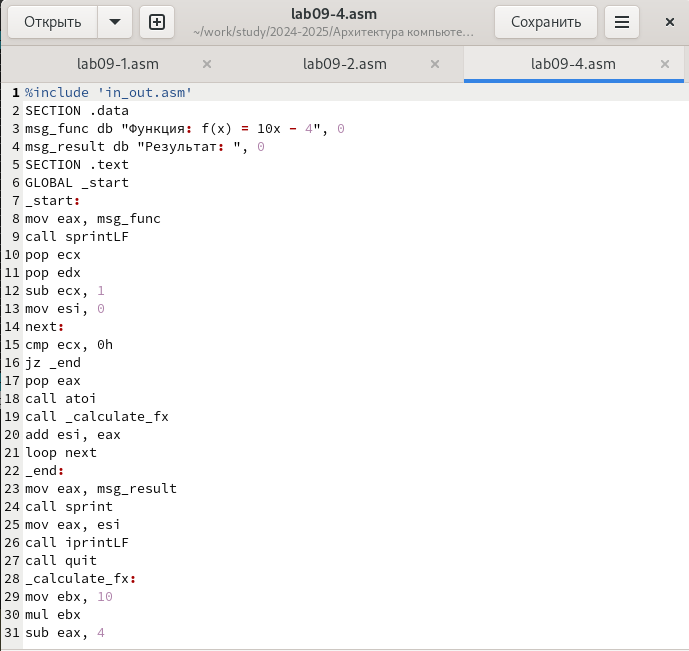


Рис. 21: Редактирование программы

Новая программа:

%include 'in\_out.asm'  
SECTION .data  
msg\_func db "Функция: f(x) = 10x - 4", 0  
msg\_result db "Результат: ", 0  
SECTION .text  
GLOBAL \_start  
\_start:  
mov eax, msg\_func  
call sprintLF  
pop ecx  
pop edx  
sub ecx, 1  
mov esi, 0  
next:  
cmp ecx, 0h  
jz \_end  
pop eax  
call atoi  
call \_calculate\_fx  
add esi, eax  
loop next  
\_end:   
mov eax, msg\_result  
call sprint  
mov eax, esi  
call iprintLF  
call quit  
\_calculate\_fx:  
mov ebx, 10  
mul ebx  
sub eax, 4

Запускаю программу в режике отладичка и пошагово через si просматриваю изменение значений регистров через i r. При выполнении инструкции mul ecx можно заметить, что результат умножения записывается в регистр eax, но также меняет и edx. Значение регистра ebx не обновляется напрямую, поэтому результат программа неверно подсчитывает функцию (рис. 22).

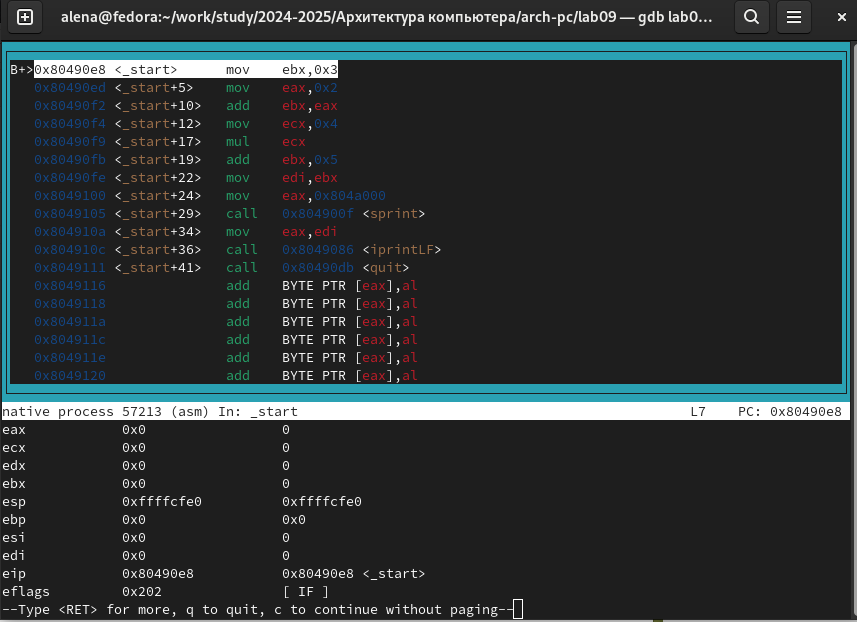


Рис. 22: Просмотр программы

Исправляю найденную ошибку, теперь программа верно считает значение функции (рис. 23).

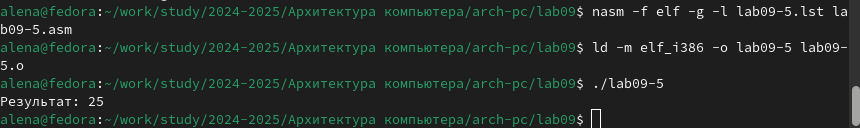


Рис. 23: Запуск исправленной программы

Новая программа:

%include 'in\_out.asm'  
  
SECTION .data  
div: DB 'Результат: ', 0  
  
SECTION .text  
GLOBAL \_start  
\_start:  
  
mov ebx, 3  
mov eax, 2  
add ebx, eax  
mov eax, ebx  
mov ecx, 4  
mul ecx  
add eax, 5  
mov edi, eax  
  
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. Ознакомилась с методами отладки при помощи GDB и его основными возможностями.

# Список литературы

1. [Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/2089096/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%969.%20%D0%9F%D0%BE%D0%BD%D1%8F%D1%82%D0%B8%D0%B5%20%D0%BF%D0%BE%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B.%20%D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D1%87%D0%B8%D0%BA%20..pdf)