Отчёт по лабораторной работе №9

Дисциплина: Архитектура компьютера

Гасанова Шакира Чингизовна

Содержание

1 Цель работы ..........................................................................................………………………………................ 3

2 Задание ........................................................................................................…………………………………......... 4

3 Теоретическое введение ...................................................................…………………………................. 5

4 Выполнение лабораторной работы

4.1 Реализация подпрограмм в NASM..............................................………………….........................6

4.2 Отладка программ с помощью GDB..............................................……………………..................9

4.3 Добавление точек останова...............................................................…………………….................14

4.4 Работа с данными программы в GDB....................................................……………….............16

4.5 Обработка аргументов командной строки в GDB...........................………….................20

4.6 Задания для самостоятельной работы...............................................…………….................22

5 Выводы ........................................................................................................…………………………………........27

6 Источники .....................................................................................................……………………………….......28

# 1 Цель работы

Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

# 2 Задание

1. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Самостоятельное выполнение заданий по материалам лабораторной работы

# 3 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа: 1. обнаружение ошибки; 2. поиск её местонахождения; 3. определение причины ошибки; 4. исправление ошибки. Можно выделить следующие типы ошибок: 1. синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; 2. семантические ошибки — являются логическими и приводят к тому, что программа запускается, отрабатывает, но не даёт желаемого результата; 3. ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль). Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно 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 (рис. 1).

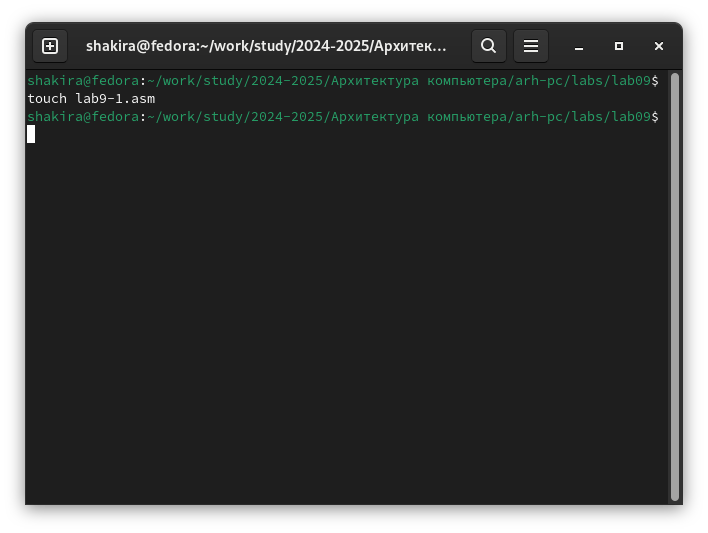


Рис. 1: Название рисунка

Копирую в файл код из листинга, компилирую и запускаю его, данная программа выполняет вычисление функции (рис. 2).

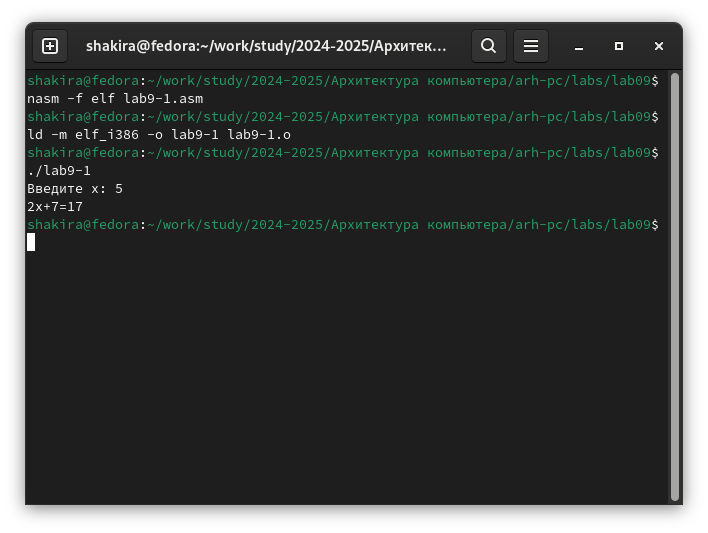


Рис. 2: 2

Изменяю текст программы, добавив в нее подпрограмму, теперь она вычисляет значение функции для выражения f(g(x)) (рис. 3).

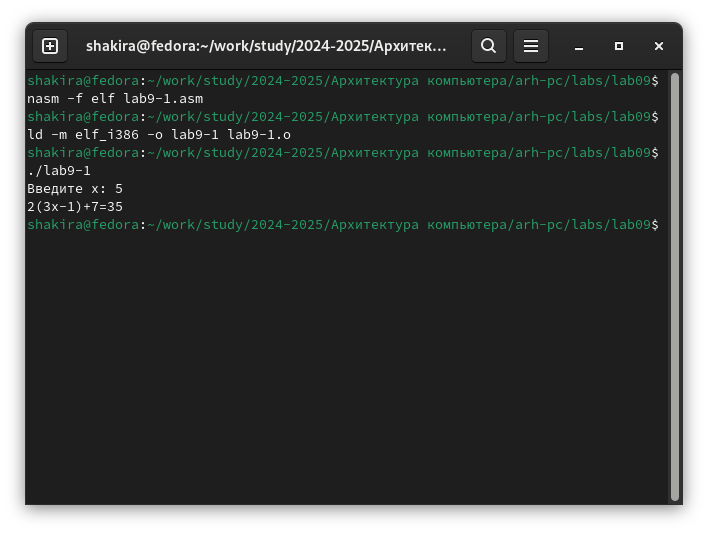


Рис. 3: 3

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 программ с помощью GDB** В созданный файл копирую программу второго листинга, транслирую с созданием файла листинга и отладки, компоную и запускаю в отладчике (рис. 4).

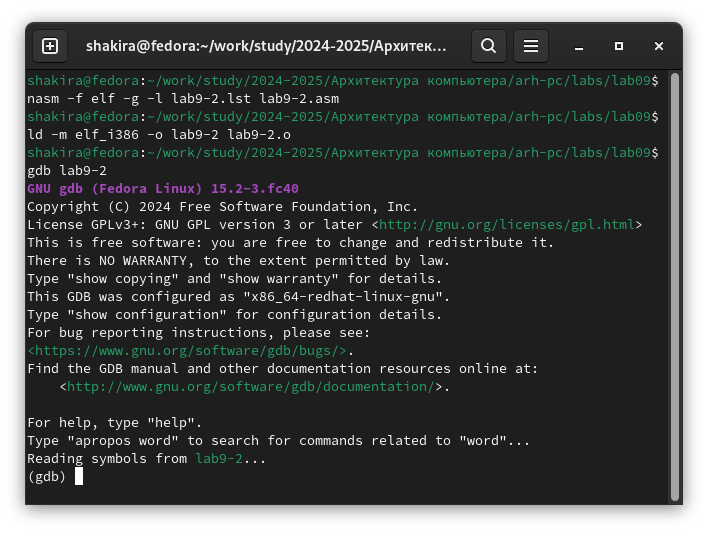


Рис. 4: 4

Запустив программу командой run, я убедилась, что она работает исправно (рис. 5).

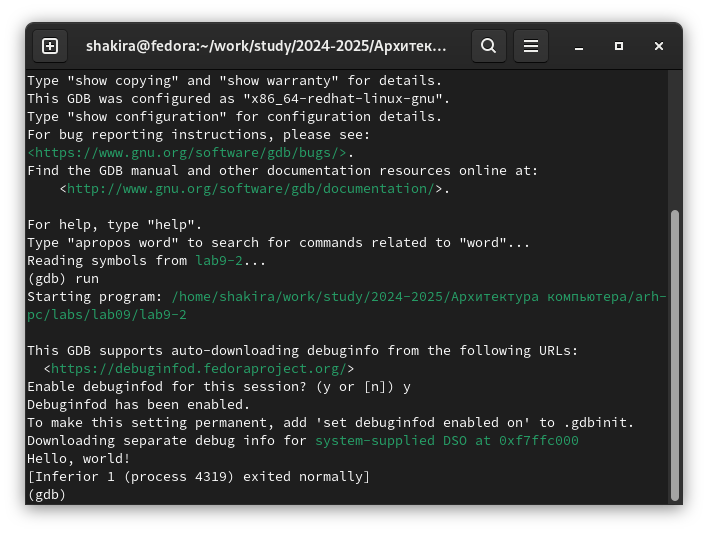


Рис. 5: 5

Для более подробного анализа программы добавляю брейкпоинт на метку \_start и снова запускаю отладку (рис. 6).

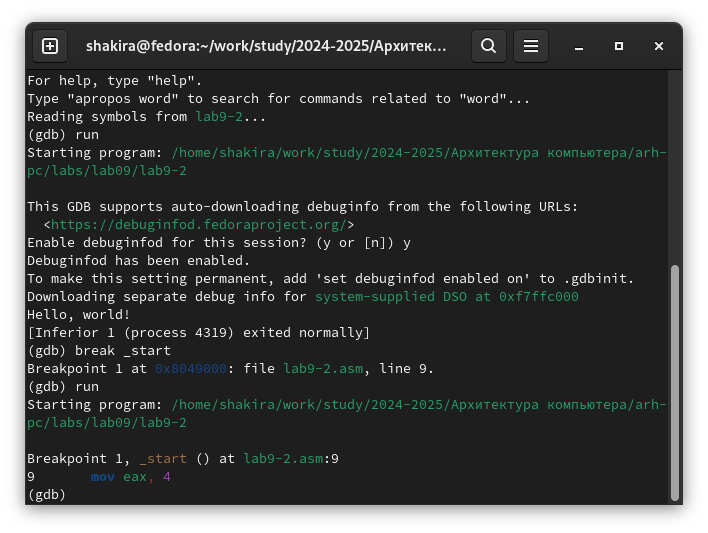


Рис. 6: 6

Когда изучаю дизассемблированный код программы, перевожу команды на синтаксис Intel для процессоров AMD. Основные различия между синтаксисами ATT и Intel следующие: 1. Порядок операндов: - ATT: Сначала указывается источник, затем назначение. - Intel: Сначала указывается назначение, затем источник. 2. Размер операндов: - ATT: Размер операндов явно указывается с помощью суффиксов, например, movl для операций с длинными данными. Непосредственные операнды (литералы) предваряются символом $. - Intel: Размер операндов определяется из контекста (например, ax, eax). Непосредственные операнды записываются без дополнительных символов. 3. Имена регистров: - ATT: Имена регистров начинаются с символа % (например, %eax). - Intel: Имена регистров записываются без префиксов (например, eax). (рис. 7).

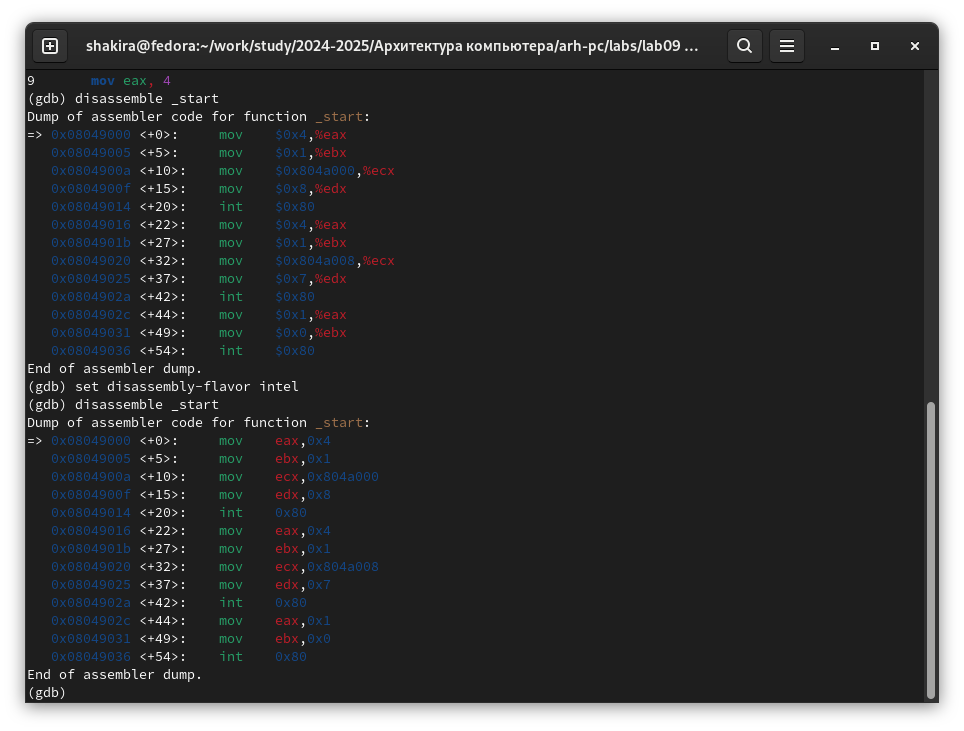


Рис. 7: 7

Включаю режим псевдографики для более удобного анализа программы (рис. 8).

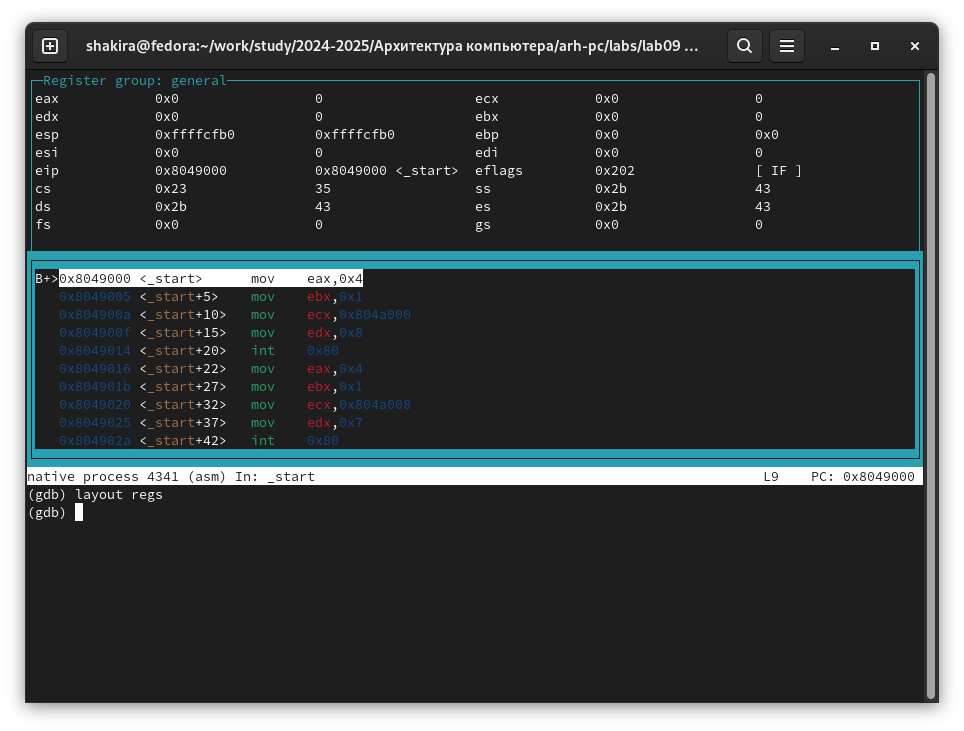


Рис. 8: 8

**4.3 Добавление точек останова** Проверяю в режиме псевдографики, что брейкпоинт сохранился (рис. 9).

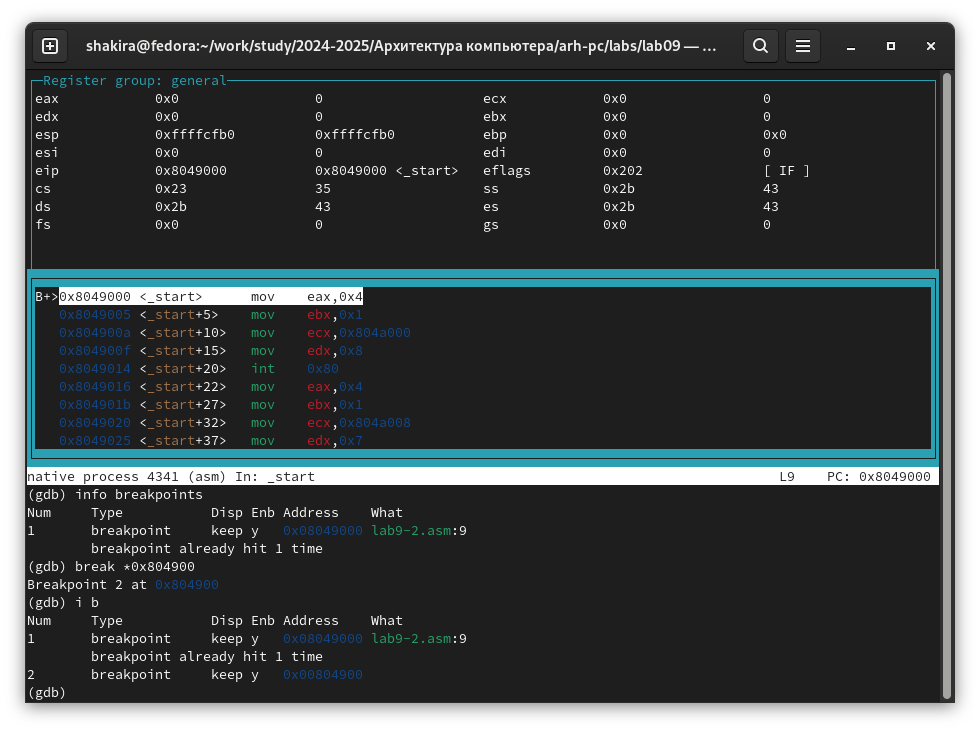


Рис. 9: 9

Устаналиваю еще одну точку останова по адресу инструкции (рис. 10).

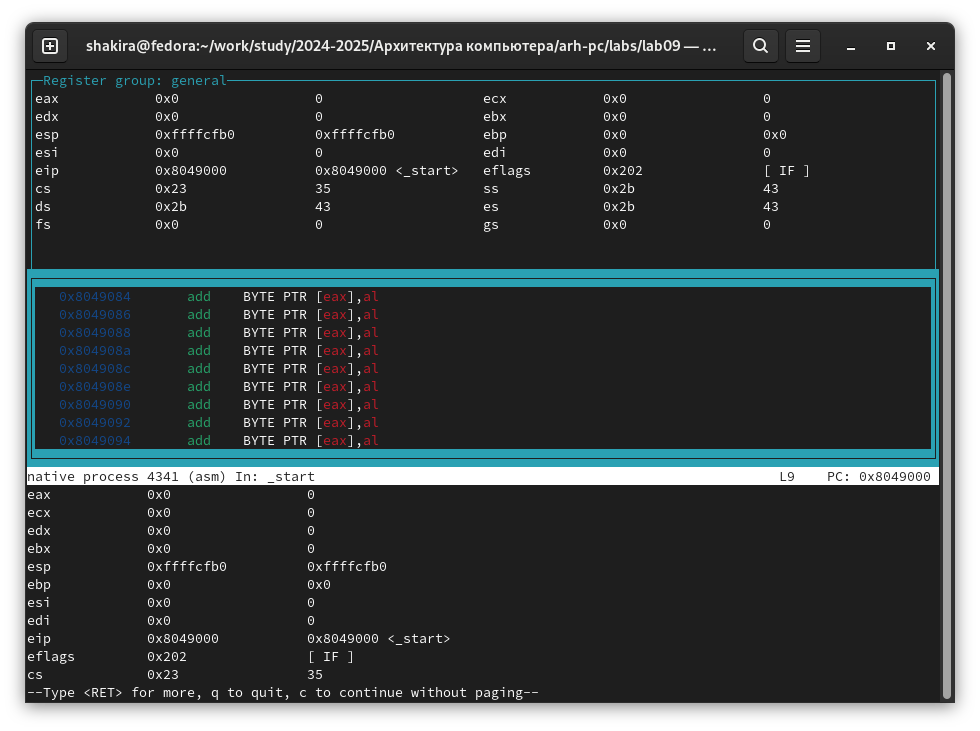


Рис. 10: 10

**4.4 Работа с данными программы в GDB** Просматриваю содержимое регистров командой info registers и смотрю содержимое переменных по имени и по адресу (рис. 11).

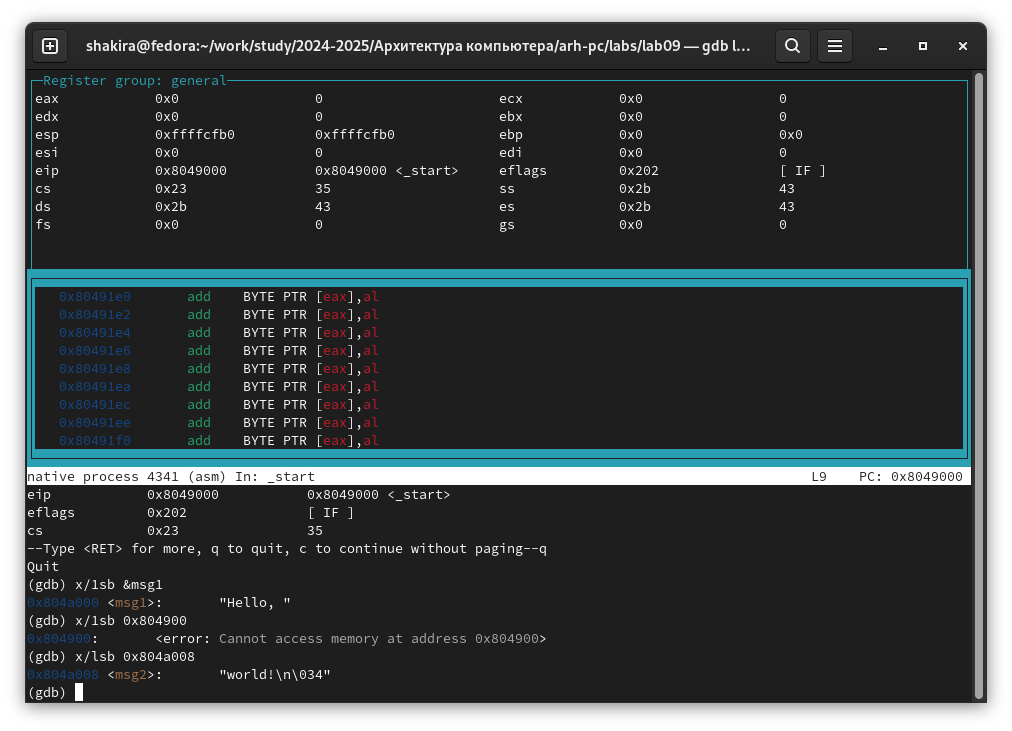


Рис. 11: 11

Меняю содержимое переменных по имени и по адресу (рис. 12).

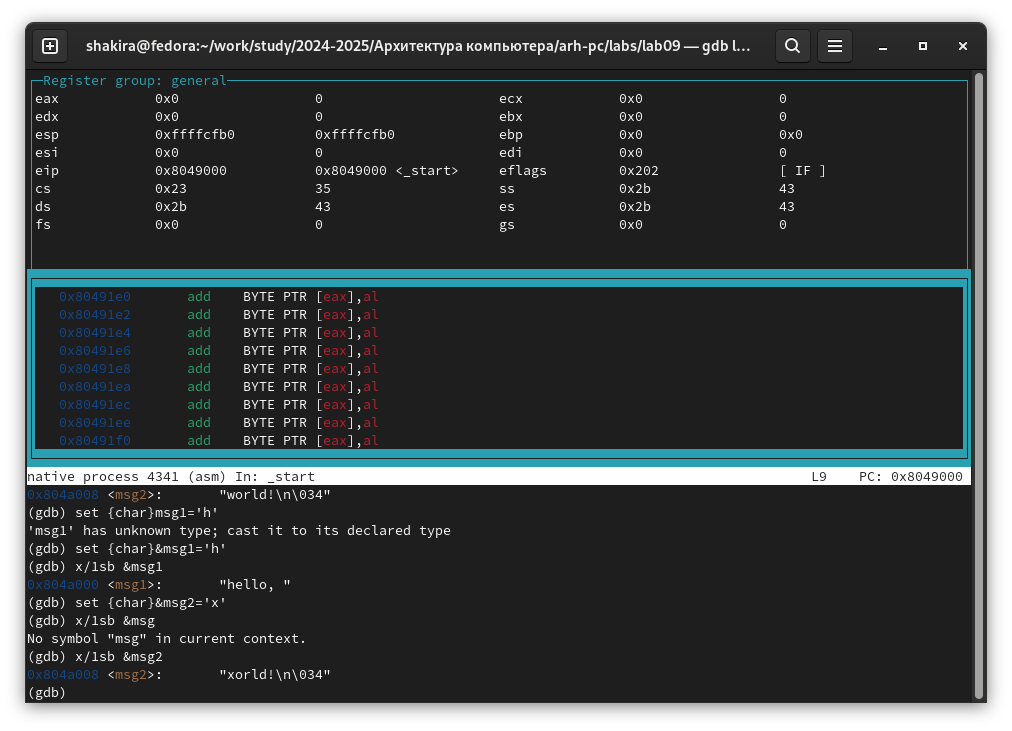


Рис. 12: 12

Вывожу в различных форматах значение регистра edx (рис. 13).

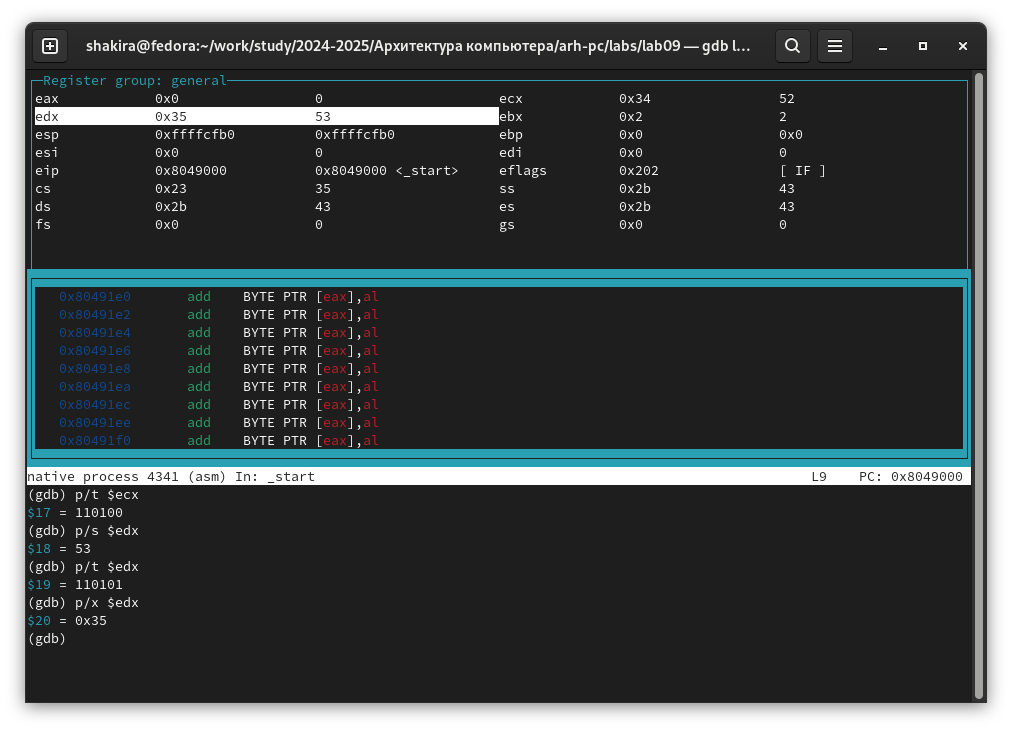


Рис. 13: 13

С помощью команды set меняю содержимое регистра ebx (рис. 14).

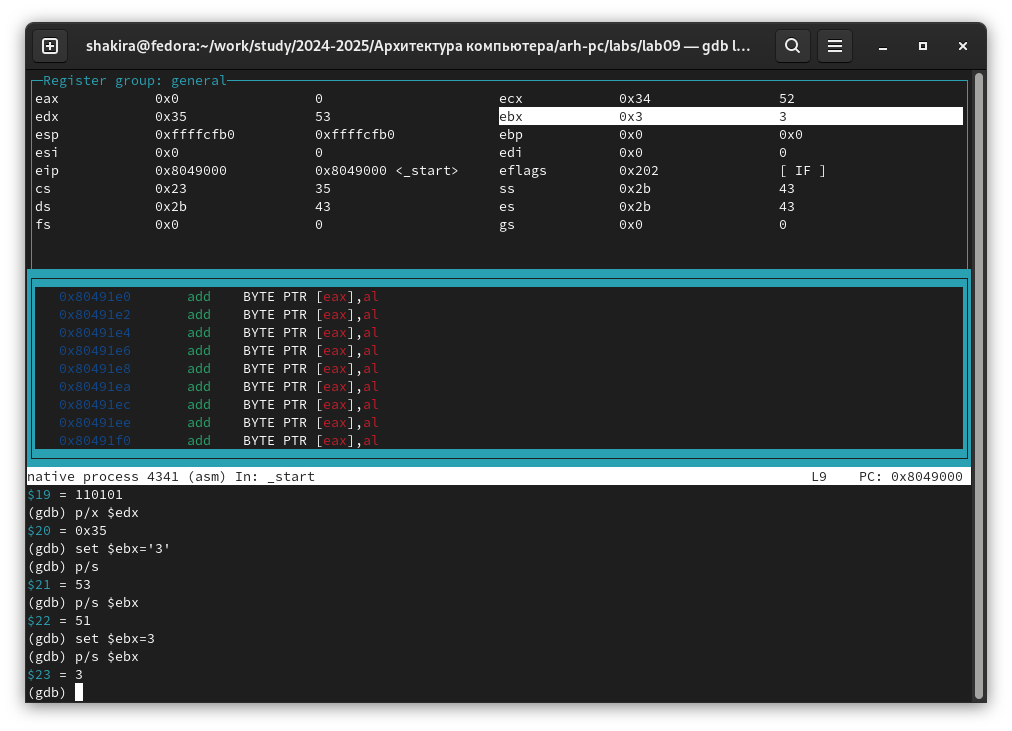


Рис. 14: 14

**4.5 Обработка аргументов командной строки в GDB** Копирую программу из предыдущей лабораторной работы в текущий каталог и создаю исполняемый файл с файлом листинга и отладки (рис. 15).

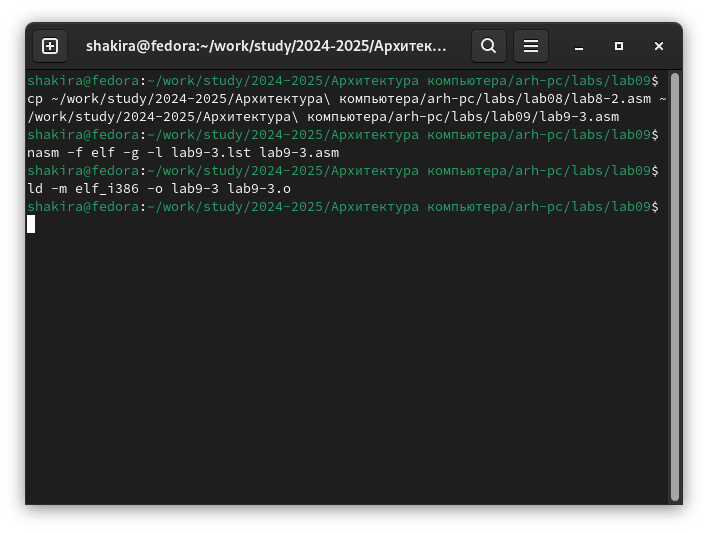


Рис. 15: 15

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. 16).

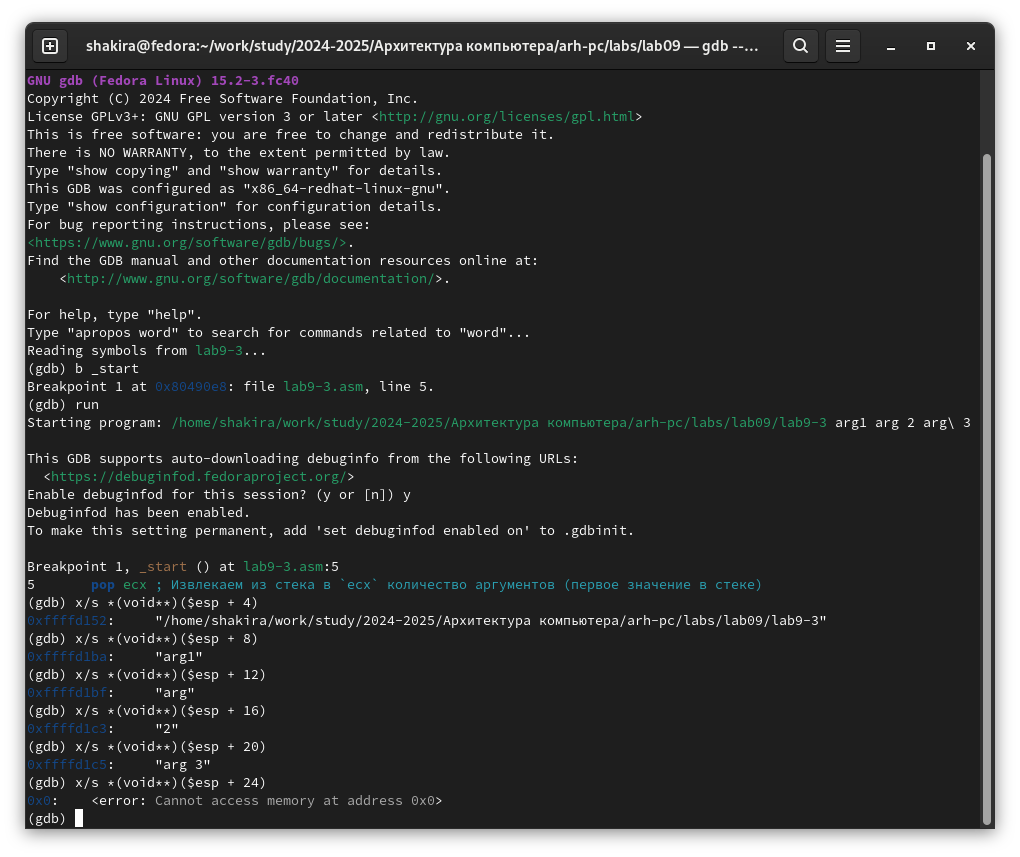
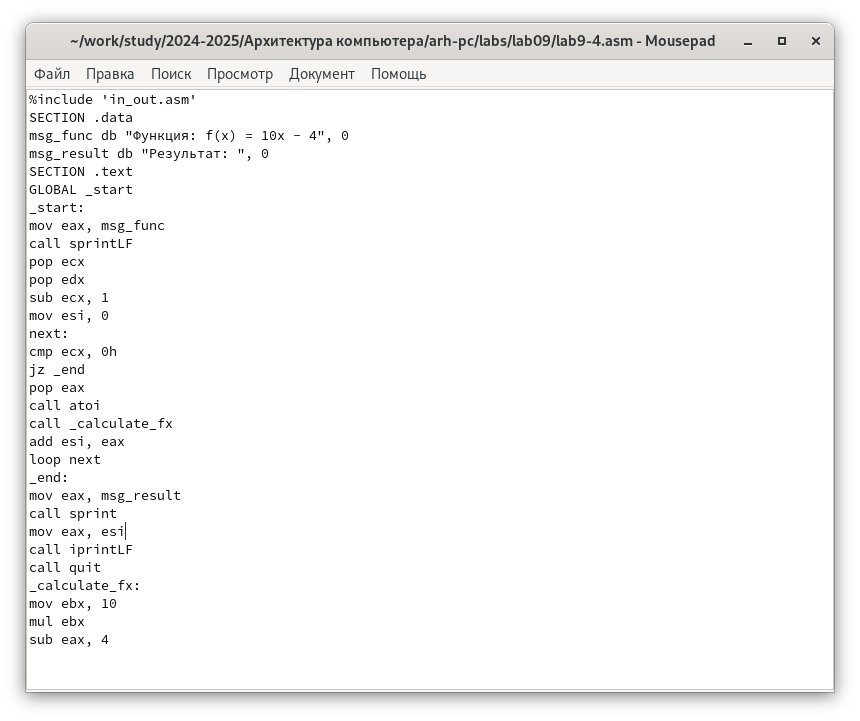


Рис. 16: 16

**4.6 Задания для самостоятельной работы** 1. Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. **¿fig:017?**).

Рис. 17: 17

Код программы:

%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

1. 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. 18).

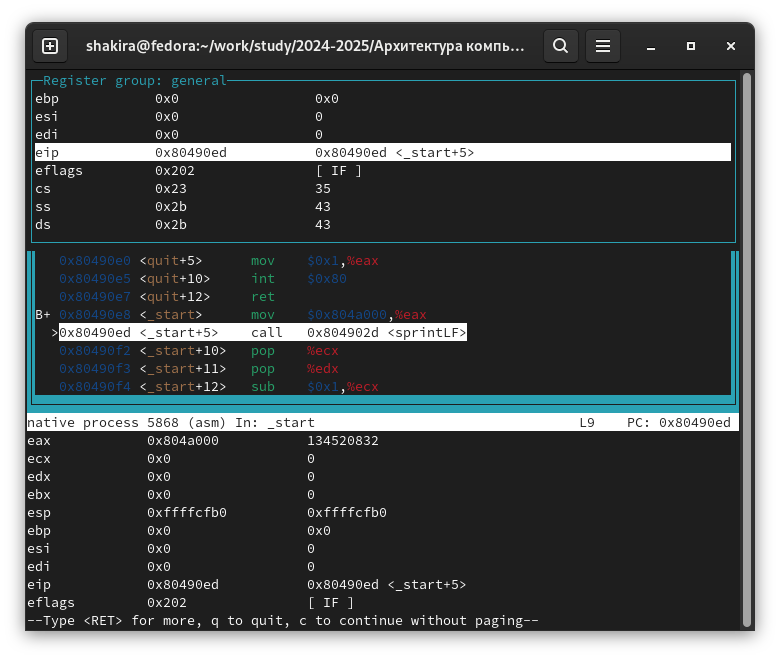


Рис. 18: 18

Исправляю найденную ошибку, теперь программа верно считает значение функции а (рис. 19).

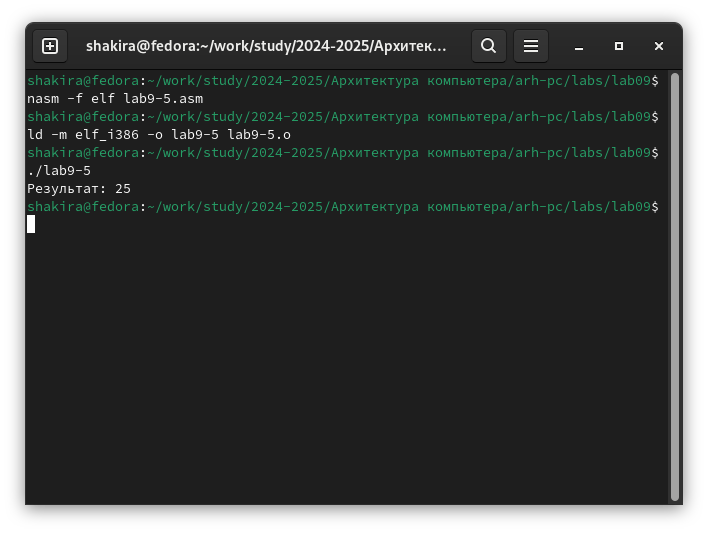


Рис. 19: 19

Код изменённой программы:

%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

В результате выполнения данной лабораторной работы я приобрела навыки написания программ с использованием подпрограмм, а также познакомилась с методами отладки при поомщи GDB и его основными возможностями.

# Список литературы

1. [Архитектура ЭВМ (rudn.ru)](https://esystem.rudn.ru/)