

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ

### ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: Архитектура компьютера

Студент: Гасанова Шакира Чингизовна

Группа: НКАбд-05-24

МОСКВА

2024 г.

# Содержание

<b>1 Цель работы .....</b>	<b>3</b>
<b>2 Задание .....</b>	<b>4</b>
<b>3 Теоретическое введение .....</b>	<b>5</b>
<b>4 Выполнение лабораторной работы</b>	
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
<b>5 Выводы .....</b>	<b>27</b>
<b>6 Источники .....</b>	<b>28</b>

## **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, отработывает, 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).

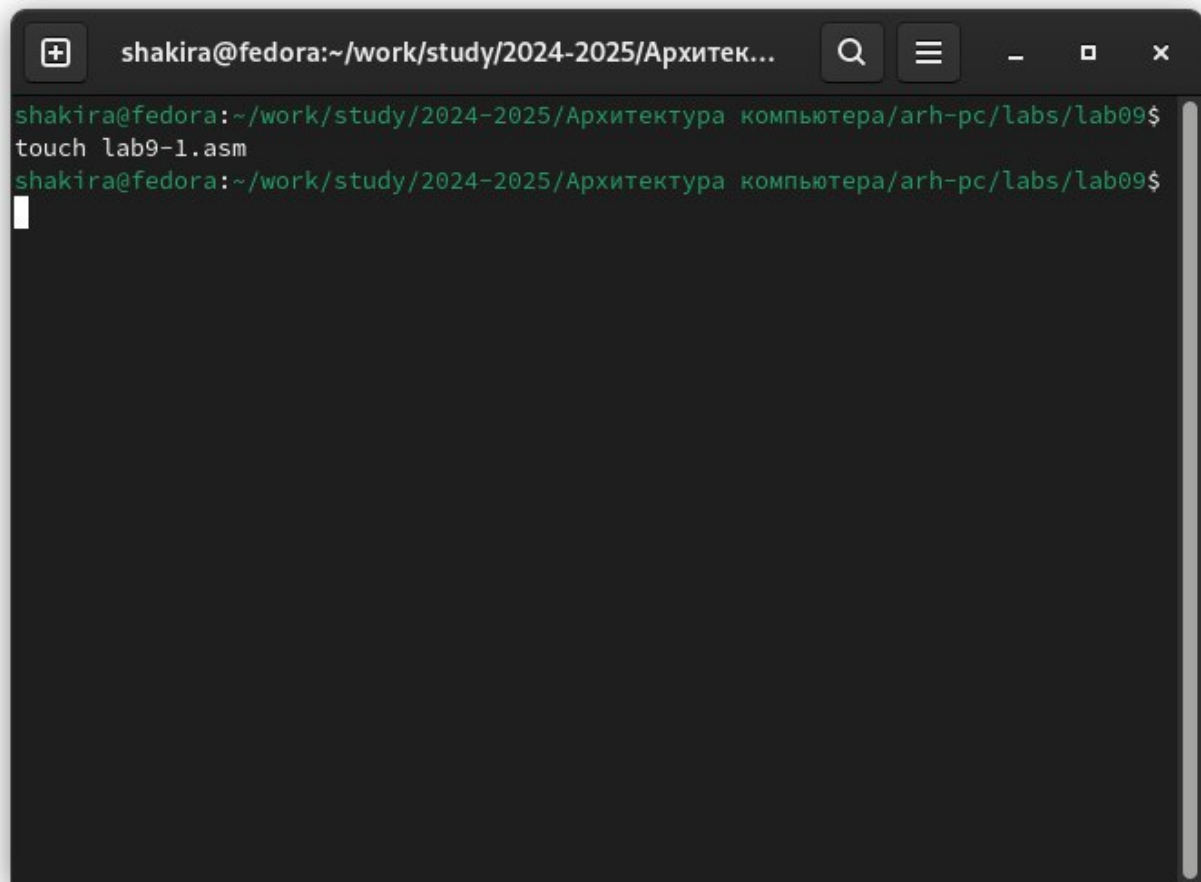
Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга.

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

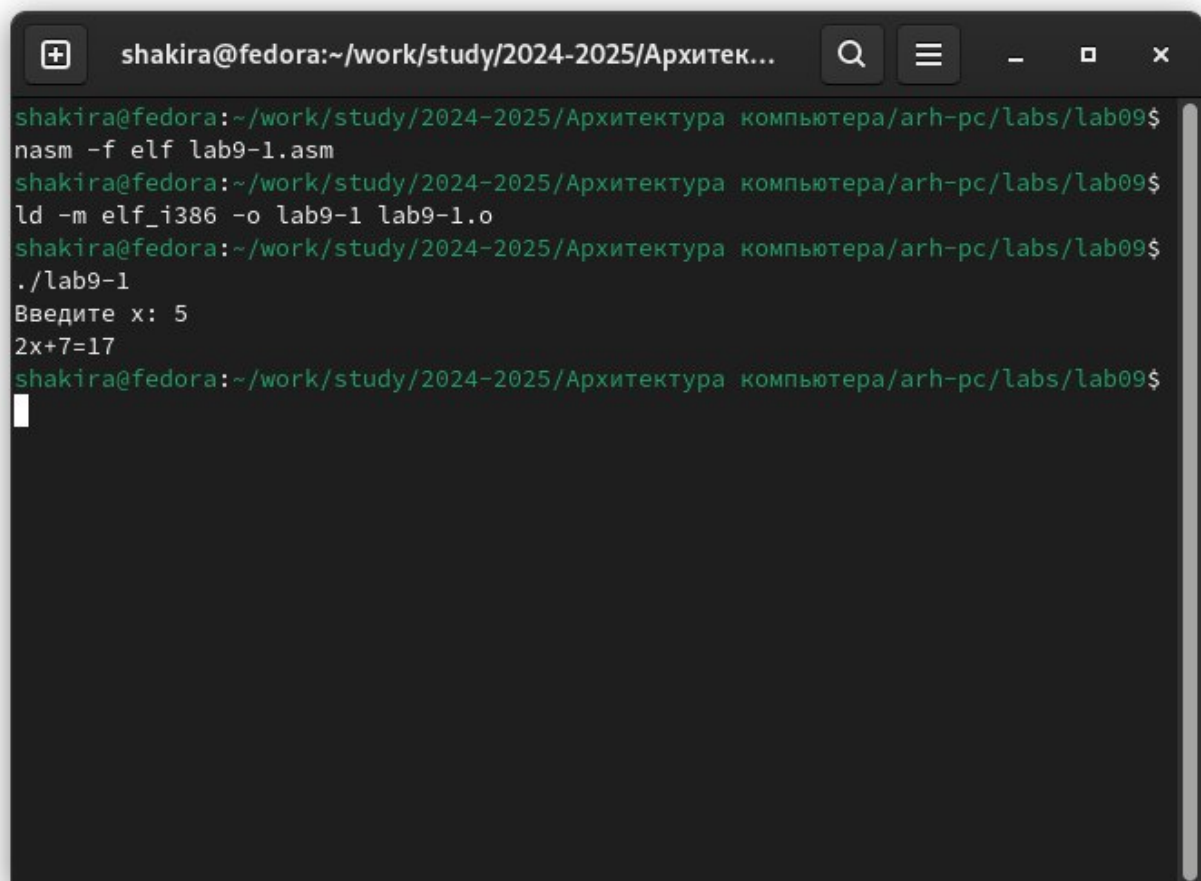
Создаю каталог для выполнения лабораторной работы №9 (рис. 1).



```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$  
touch lab9-1.asm  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$
```

Рис.1 Создание каталога

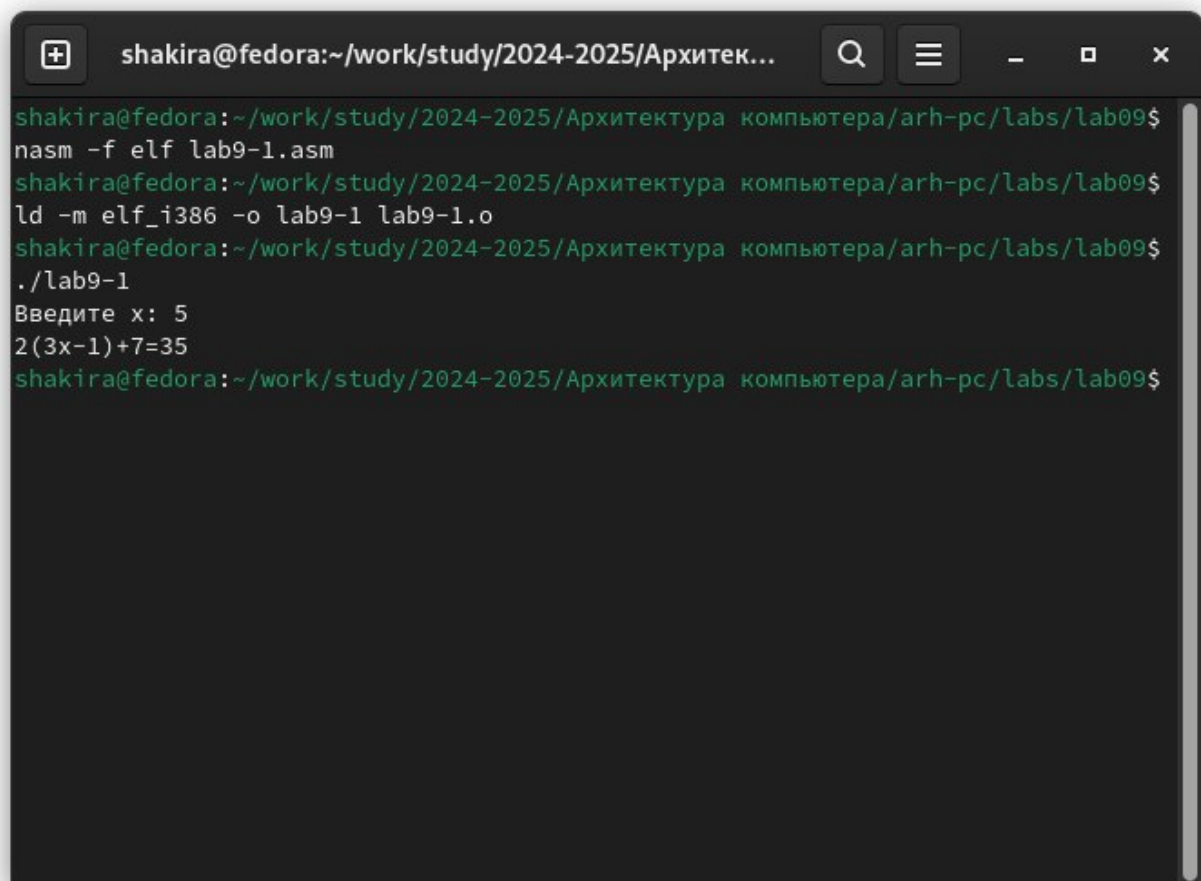
Копирую в файл код из листинга, компилирую и запускаю его, данная программа выполняет вычисление функции (рис. 2).

A terminal window with a dark background and light green text. The window title is "shakira@fedora:~/work/study/2024-2025/Архитек...". The terminal shows the following commands and output:

```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$  
nasm -f elf lab9-1.asm  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$  
ld -m elf_i386 -o lab9-1 lab9-1.o  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$  
./lab9-1  
Введите x: 5  
2x+7=17  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$
```

Рис.2 Запуск программы

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

A terminal window with a dark background and light green text. The title bar shows the user 'shakira' on a 'fedora' machine, with the current directory being '~ /work/study/2024-2025/Архитек...'. The terminal shows the following commands and output:

```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$  
nasm -f elf lab9-1.asm  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$  
ld -m elf_i386 -o lab9-1 lab9-1.o  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$  
./lab9-1  
Введите x: 5  
2(3x-1)+7=35  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$
```

Рис.3 Редактирование программы

Код программы:

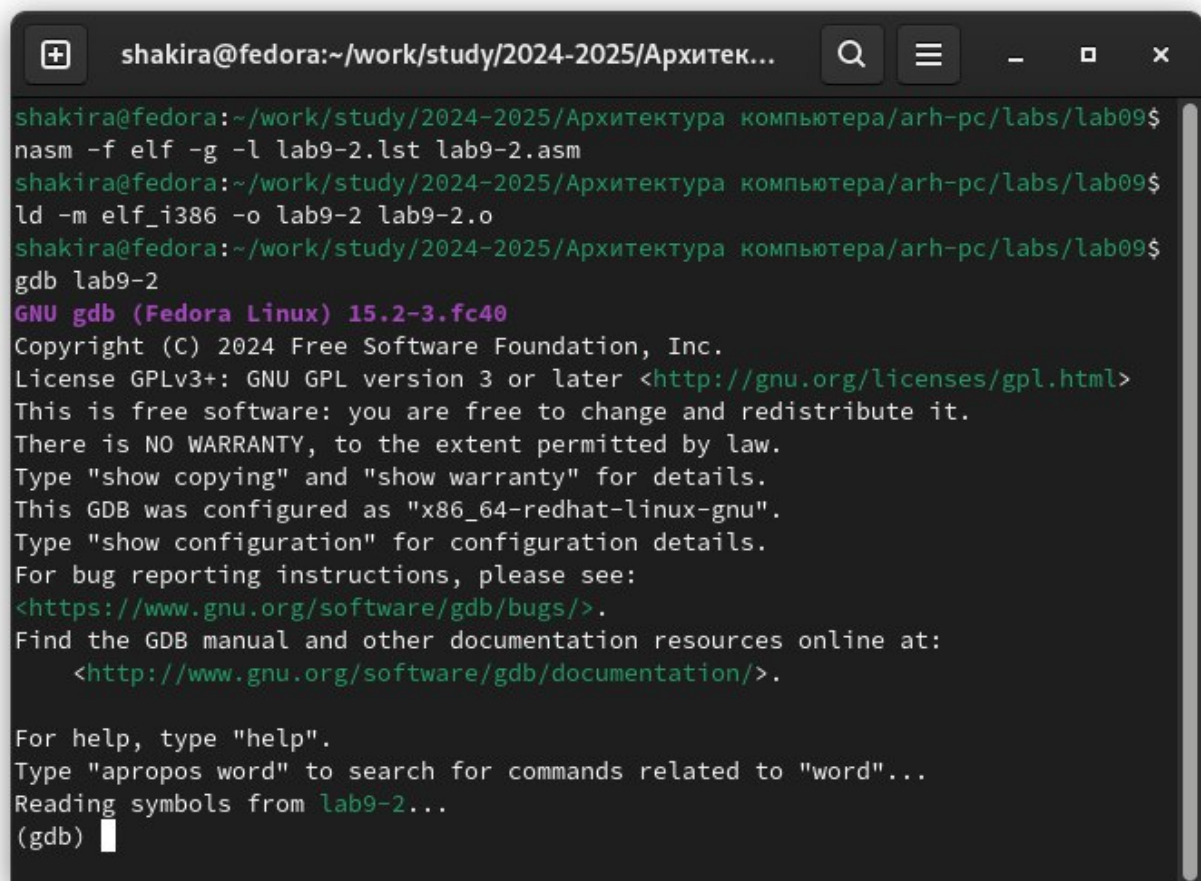
```
%include 'in_out.asm'  
SECTION .data  
msg: DB 'Введите x: ', 0  
result: DB '2(3x-1)+7=', 0  
SECTION .bss  
x: RESB 80  
res: RESB 80  
SECTION .text  
GLOBAL _start  
_start:  
mov eax, msg  
call sprint  
mov ecx, x  
mov edx, 80
```



```
call sread
mov eax, x
call atoi
call _calcul
mov eax, result
call sprint
mov eax, [res]
call iprintLF
10
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).

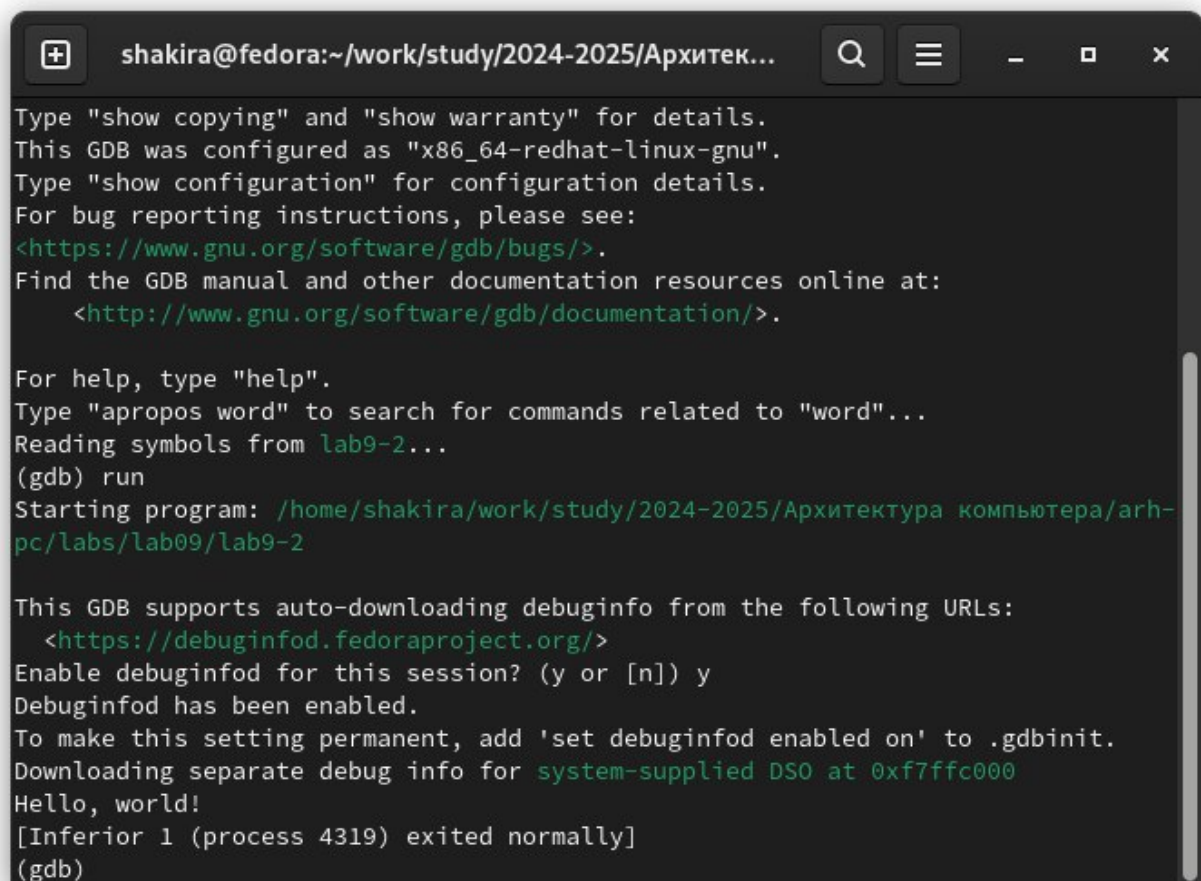


```
shakira@fedora:~/work/study/2024-2025/Архитек...
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$
nasm -f elf -g -l lab9-2.lst lab9-2.asm
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$
ld -m elf_i386 -o lab9-2 lab9-2.o
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$
gdb lab9-2
GNU gdb (Fedora Linux) 15.2-3.fc40
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) 
```

Рис.4 Запуск программы в отладчике

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



The image shows a terminal window with a dark background. The title bar at the top reads "shakira@fedora:~/work/study/2024-2025/Архитек...". The terminal content shows the GDB startup sequence: it displays introductory text about GDB, the user enters "run", the program starts and prints "Hello, world!", and finally, the program exits normally. The user's prompt "(gdb)" is visible at the end of the last line.

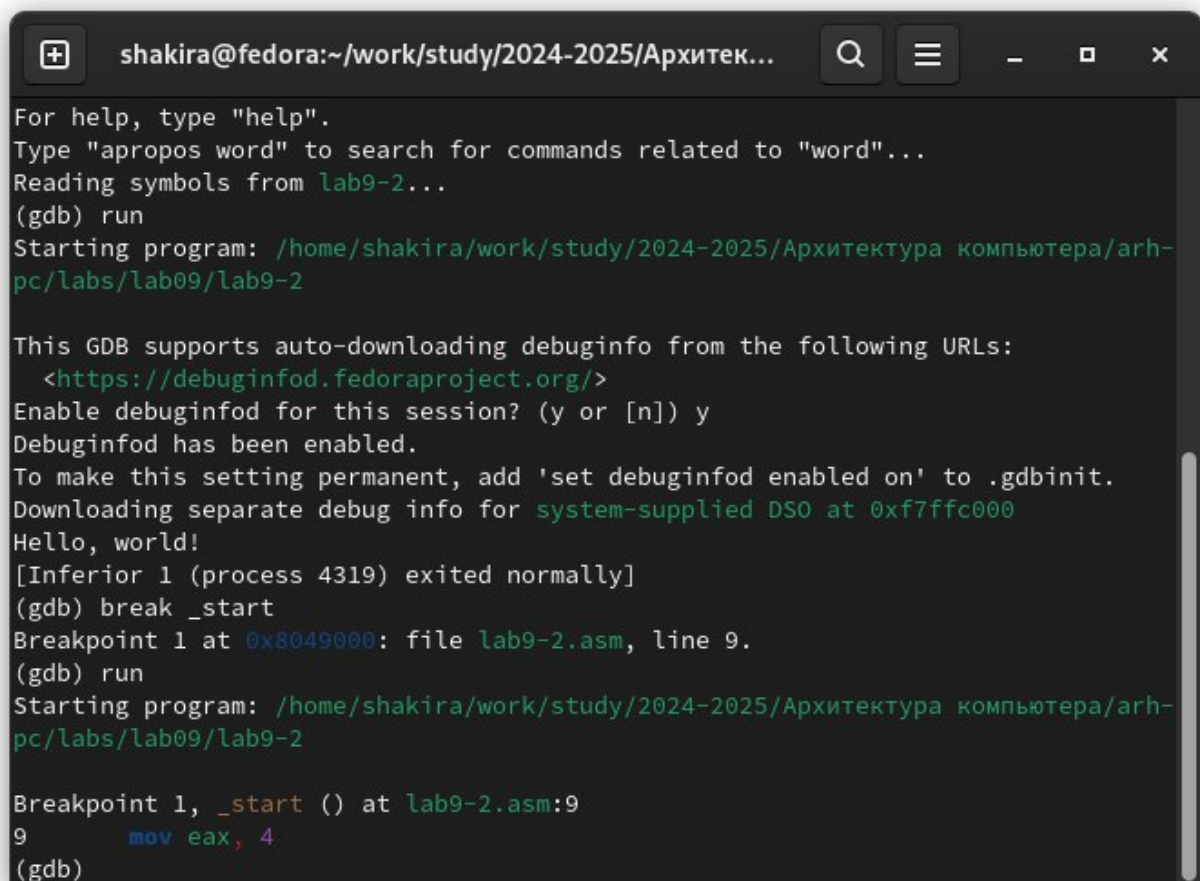
```
shakira@fedora:~/work/study/2024-2025/Архитек...
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) run
Starting program: /home/shakira/work/study/2024-2025/Архитектура компьютера/arh-
pc/labs/lab09/lab9-2

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 4319) exited normally]
(gdb)
```

Рис.5 Проверка программы отладчиком

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



```
shakira@fedora:~/work/study/2024-2025/Архитек...
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) run
Starting program: /home/shakira/work/study/2024-2025/Архитектура компьютера/arh-
pc/labs/lab09/lab9-2

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 4319) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/shakira/work/study/2024-2025/Архитектура компьютера/arh-
pc/labs/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
(gdb)
```

Рис.6 Запуск отладчика с брейкпоинтом

Когда изучаю дизассемблированный код программы, перевожу команды на синтаксис Intel для процессоров AMD. Основные различия между синтаксисами АТТ и Intel следующие:

1. Порядок операндов:

- АТТ: Сначала указывается источник, затем назначение.
- Intel: Сначала указывается назначение, затем источник.

2. Размер операндов:

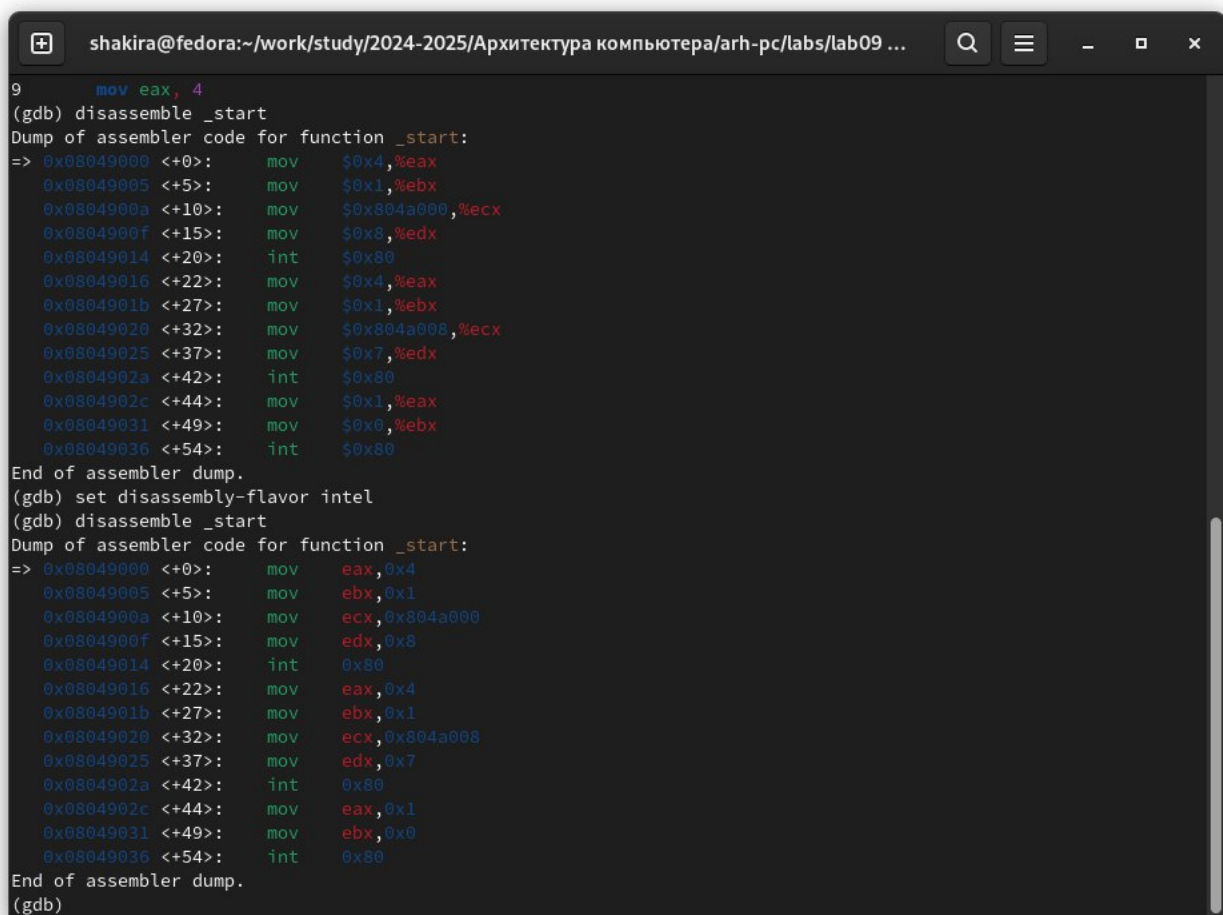
- АТТ: Размер операндов явно указывается с помощью суффиксов, например, `movl` для операций с длинными данными. Непосредственные операнды (литералы) предваряются символом `$`.

- Intel: Размер операндов определяется из контекста (например, `ax`, `eax`). Непосредственные операнды записываются без дополнительных символов.

3. Имена регистров:

- АТТ: Имена регистров начинаются с символа `%` (например, `%eax`).
- Intel: Имена регистров записываются без префиксов (например, `eax`).

(рис. 7).



```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09 ...
9      mov eax, 4
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:    mov     $0x4,%eax
      0x08049005 <+5>:    mov     $0x1,%ebx
      0x0804900a <+10>:   mov     $0x804a000,%ecx
      0x0804900f <+15>:   mov     $0x8,%edx
      0x08049014 <+20>:   int     $0x80
      0x08049016 <+22>:   mov     $0x4,%eax
      0x0804901b <+27>:   mov     $0x1,%ebx
      0x08049020 <+32>:   mov     $0x804a008,%ecx
      0x08049025 <+37>:   mov     $0x7,%edx
      0x0804902a <+42>:   int     $0x80
      0x0804902c <+44>:   mov     $0x1,%eax
      0x08049031 <+49>:   mov     $0x0,%ebx
      0x08049036 <+54>:   int     $0x80
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:    mov     eax,0x4
      0x08049005 <+5>:    mov     ebx,0x1
      0x0804900a <+10>:   mov     ecx,0x804a000
      0x0804900f <+15>:   mov     edx,0x8
      0x08049014 <+20>:   int     0x80
      0x08049016 <+22>:   mov     eax,0x4
      0x0804901b <+27>:   mov     ebx,0x1
      0x08049020 <+32>:   mov     ecx,0x804a008
      0x08049025 <+37>:   mov     edx,0x7
      0x0804902a <+42>:   int     0x80
      0x0804902c <+44>:   mov     eax,0x1
      0x08049031 <+49>:   mov     ebx,0x0
      0x08049036 <+54>:   int     0x80
End of assembler dump.
(gdb)
```

Рис.7 Дисассимилирование программы

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

```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09 ...
Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcfb0 0xffffcfb0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049000 0x8049000 <_start>  eflags   0x202    [ IF ]
cs       0x23     35     ss       0x2b     43
ds       0x2b     43     es       0x2b     43
fs       0x0      0      gs       0x0      0

B> 0x8049000 <_start>  mov    eax, 0x4
0x8049005 <_start+5>  mov    ebx, 0x1
0x804900a <_start+10> mov    ecx, 0x804a000
0x804900f <_start+15> mov    edx, 0x8
0x8049014 <_start+20> int    0x80
0x8049016 <_start+22> mov    eax, 0x4
0x804901b <_start+27> mov    ebx, 0x1
0x8049020 <_start+32> mov    ecx, 0x804a008
0x8049025 <_start+37> mov    edx, 0x7
0x804902a <_start+42> int    0x80

native process 4341 (asm) In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb) |
```

Рис.8 Режим псевдографики

### 4.3 Добавление точек останова

Проверяю в режиме псевдографики, что брейкпоинт сохранился (рис. 9).

```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09 — ...

Register group: general
eax    0x0          0          ecx    0x0          0
edx    0x0          0          ebx    0x0          0
esp    0xffffcfb0   0xffffcfb0  ebp    0x0          0x0
esi    0x0          0          edi    0x0          0
eip    0x8049000    0x8049000  <_start>  eflags  0x202        [ IF ]
cs     0x23         35         ss     0x2b         43
ds     0x2b         43         es     0x2b         43
fs     0x0          0          gs     0x0          0

B> 0x8049000 <_start> mov    eax,0x4
0x8049005 <_start+5> mov    ebx,0x1
0x804900a <_start+10> mov    ecx,0x804a000
0x804900f <_start+15> mov    edx,0x8
0x8049014 <_start+20> int    0x80
0x8049016 <_start+22> mov    eax,0x4
0x804901b <_start+27> mov    ebx,0x1
0x8049020 <_start+32> mov    ecx,0x804a008
0x8049025 <_start+37> mov    edx,0x7

native process 4341 (asm) In: _start L9 PC: 0x8049000
(gdb) info breakpoints
Num    Type          Disp Enb Address      What
1      breakpoint    keep y  0x08049000 lab9-2.asm:9
      breakpoint already hit 1 time
(gdb) break *0x804900
Breakpoint 2 at 0x804900
(gdb) i b
Num    Type          Disp Enb Address      What
1      breakpoint    keep y  0x08049000 lab9-2.asm:9
      breakpoint already hit 1 time
2      breakpoint    keep y  0x08049000
(gdb)
```

Рис.9 Список брейкпоинтов

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



```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09 — ...
Register group: general
eax    0x0      0      ecx    0x0      0
edx    0x0      0      ebx    0x0      0
esp    0xffffcfb0 0xffffcfb0 ebp    0x0      0x0
esi    0x0      0      edi    0x0      0
eip    0x8049000 0x8049000 <_start> eflags 0x202    [ IF ]
cs     0x23     35     ss     0x2b     43
ds     0x2b     43     es     0x2b     43
fs     0x0      0      gs     0x0      0

0x8049084 add BYTE PTR [eax],al
0x8049086 add BYTE PTR [eax],al
0x8049088 add BYTE PTR [eax],al
0x804908a add BYTE PTR [eax],al
0x804908c add BYTE PTR [eax],al
0x804908e add BYTE PTR [eax],al
0x8049090 add BYTE PTR [eax],al
0x8049092 add BYTE PTR [eax],al
0x8049094 add BYTE PTR [eax],al

native process 4341 (asm) In: _start L9 PC: 0x8049000
eax    0x0      0
ecx    0x0      0
edx    0x0      0
ebx    0x0      0
esp    0xffffcfb0 0xffffcfb0
ebp    0x0      0x0
esi    0x0      0
edi    0x0      0
eip    0x8049000 0x8049000 <_start>
eflags 0x202    [ IF ]
cs     0x23     35
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис.10 Добавление второй точки останова

## 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).



```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09 — gdb L...

Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcfb0 0xffffcfb0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049000 0x8049000 <_start>  eflags    0x202    [ IF ]
cs       0x23     35     ss       0x2b     43
ds       0x2b     43     es       0x2b     43
fs       0x0      0      gs       0x0      0

0x80491e0  add  BYTE PTR [eax],al
0x80491e2  add  BYTE PTR [eax],al
0x80491e4  add  BYTE PTR [eax],al
0x80491e6  add  BYTE PTR [eax],al
0x80491e8  add  BYTE PTR [eax],al
0x80491ea  add  BYTE PTR [eax],al
0x80491ec  add  BYTE PTR [eax],al
0x80491ee  add  BYTE PTR [eax],al
0x80491f0  add  BYTE PTR [eax],al

native process 4341 (asm) In: _start L9 PC: 0x8049000
eip      0x8049000 0x8049000 <_start>
eflags    0x202    [ IF ]
cs       0x23     35
--Type <RET> for more, q to quit, c to continue without paging--q
Quit
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804900
0x804900: <error: Cannot access memory at address 0x804900>
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb)
```

Рис.11 Просмотр содержимого регистров

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

```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09 — gdb L...

Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcfb0 0xffffcfb0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049000 0x8049000 <_start>  eflags    0x202    [ IF ]
cs       0x23     35     ss       0x2b     43
ds       0x2b     43     es       0x2b     43
fs       0x0      0      gs       0x0      0

0x80491e0  add  BYTE PTR [eax],al
0x80491e2  add  BYTE PTR [eax],al
0x80491e4  add  BYTE PTR [eax],al
0x80491e6  add  BYTE PTR [eax],al
0x80491e8  add  BYTE PTR [eax],al
0x80491ea  add  BYTE PTR [eax],al
0x80491ec  add  BYTE PTR [eax],al
0x80491ee  add  BYTE PTR [eax],al
0x80491f0  add  BYTE PTR [eax],al

native process 4341 (asm) In: _start L9 PC: 0x8049000
0x804a008 <msg2>: "world!\n\034"
(gdb) set {char}msg1='h'
'msg1' has unknown type; cast it to its declared type
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hello, "
(gdb) set {char}&msg2='x'
(gdb) x/1sb &msg
No symbol "msg" in current context.
(gdb) x/1sb &msg2
0x804a008 <msg2>: "xorld!\n\034"
(gdb)
```

Рис.12 Изменение содержимого переменных двумя способами

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

```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09 — gdb L...

Register group: general
eax      0x0      0      ecx      0x34      52
edx      0x35      53      ebx      0x2      2
esp      0xffffcfb0 0xffffcfb0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049000 0x8049000 <_start>  eflags    0x202    [ IF ]
cs       0x23      35      ss       0x2b      43
ds       0x2b      43      es       0x2b      43
fs       0x0      0      gs       0x0      0

0x80491e0  add  BYTE PTR [eax],al
0x80491e2  add  BYTE PTR [eax],al
0x80491e4  add  BYTE PTR [eax],al
0x80491e6  add  BYTE PTR [eax],al
0x80491e8  add  BYTE PTR [eax],al
0x80491ea  add  BYTE PTR [eax],al
0x80491ec  add  BYTE PTR [eax],al
0x80491ee  add  BYTE PTR [eax],al
0x80491f0  add  BYTE PTR [eax],al

native process 4341 (asm) In: _start L9 PC: 0x8049000
(gdb) p/t $ecx
$17 = 110100
(gdb) p/s $edx
$18 = 53
(gdb) p/t $edx
$19 = 110101
(gdb) p/x $edx
$20 = 0x35
(gdb)
```

Рис.13 Просмотр значения регистра разными представлениями

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

```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09 — gdb L...

Register group: general
eax      0x0      0      ecx      0x34      52
edx      0x35      53      ebx      0x3       3
esp      0xffffcfb0 0xffffcfb0 ebp      0x0       0x0
esi      0x0       0      edi      0x0       0
eip      0x8049000 0x8049000 <_start> eflags   0x202     [ IF ]
cs       0x23      35      ss       0x2b      43
ds       0x2b      43      es       0x2b      43
fs       0x0       0      gs       0x0       0

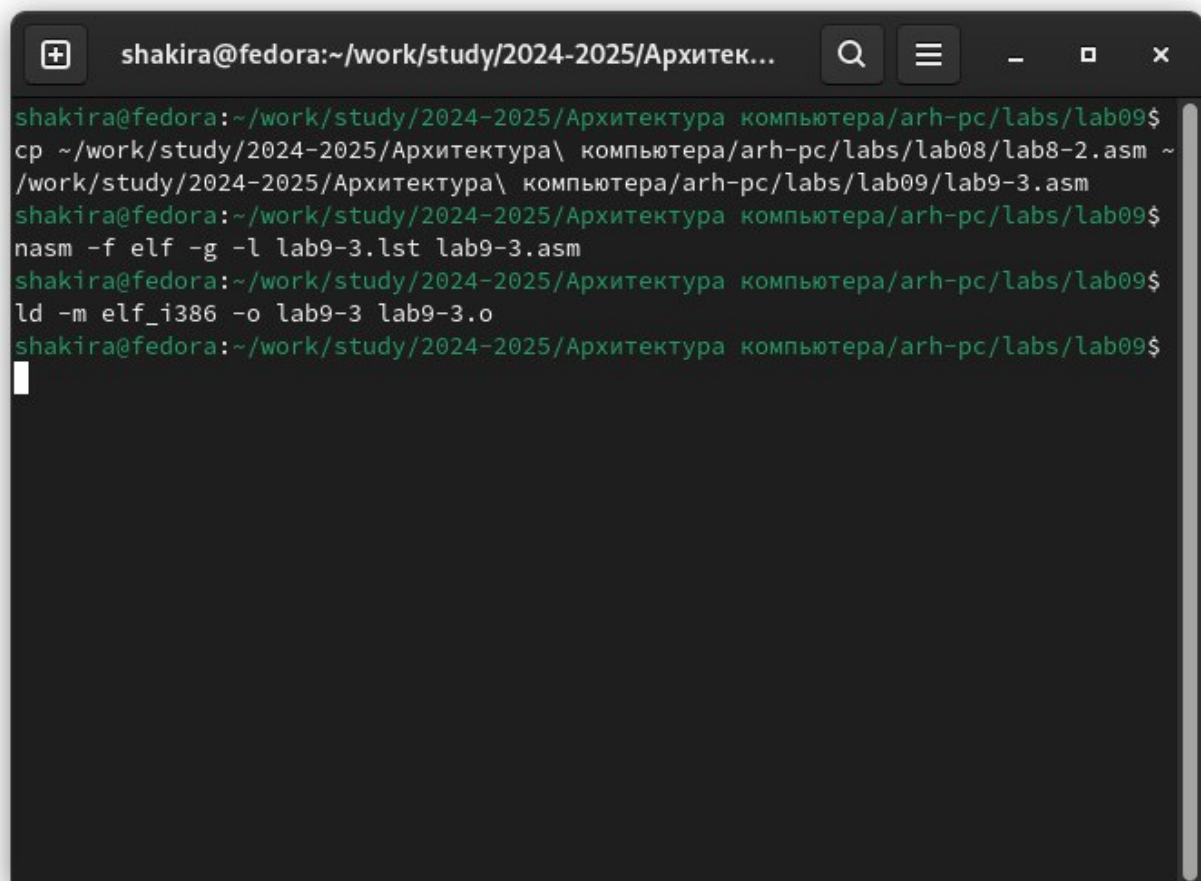
0x80491e0  add  BYTE PTR [eax],al
0x80491e2  add  BYTE PTR [eax],al
0x80491e4  add  BYTE PTR [eax],al
0x80491e6  add  BYTE PTR [eax],al
0x80491e8  add  BYTE PTR [eax],al
0x80491ea  add  BYTE PTR [eax],al
0x80491ec  add  BYTE PTR [eax],al
0x80491ee  add  BYTE PTR [eax],al
0x80491f0  add  BYTE PTR [eax],al

native process 4341 (asm) In: _start L9 PC: 0x8049000
$19 = 110101
(gdb) p/x $edx
$20 = 0x35
(gdb) set $ebx='3'
(gdb) p/s
$21 = 53
(gdb) p/s $ebx
$22 = 51
(gdb) set $ebx=3
(gdb) p/s $ebx
$23 = 3
(gdb)
```

Рис.14 Примеры использования команды set

## 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).

A terminal window with a dark background and green text. The window title is "shakira@fedora:~/work/study/2024-2025/Архитек...". The terminal shows the following commands and their outputs:

```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$  
cp ~/work/study/2024-2025/Архитектура\ компьютера/arh-pc/labs/lab08/lab8-2.asm ~  
/work/study/2024-2025/Архитектура\ компьютера/arh-pc/labs/lab09/lab9-3.asm  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$  
nasm -f elf -g -l lab9-3.lst lab9-3.asm  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$  
ld -m elf_i386 -o lab9-3 lab9-3.o  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$  
█
```

Рис.15 Подготовка новой программы

Запускаю программу с режиме отладки с указанием аргументов, указываю брейкпоинт 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).

```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09 — gdb --...
GNU gdb (Fedora Linux) 15.2-3.fc40
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) run
Starting program: /home/shakira/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09/lab9-3 arg1 arg 2 arg\ 3

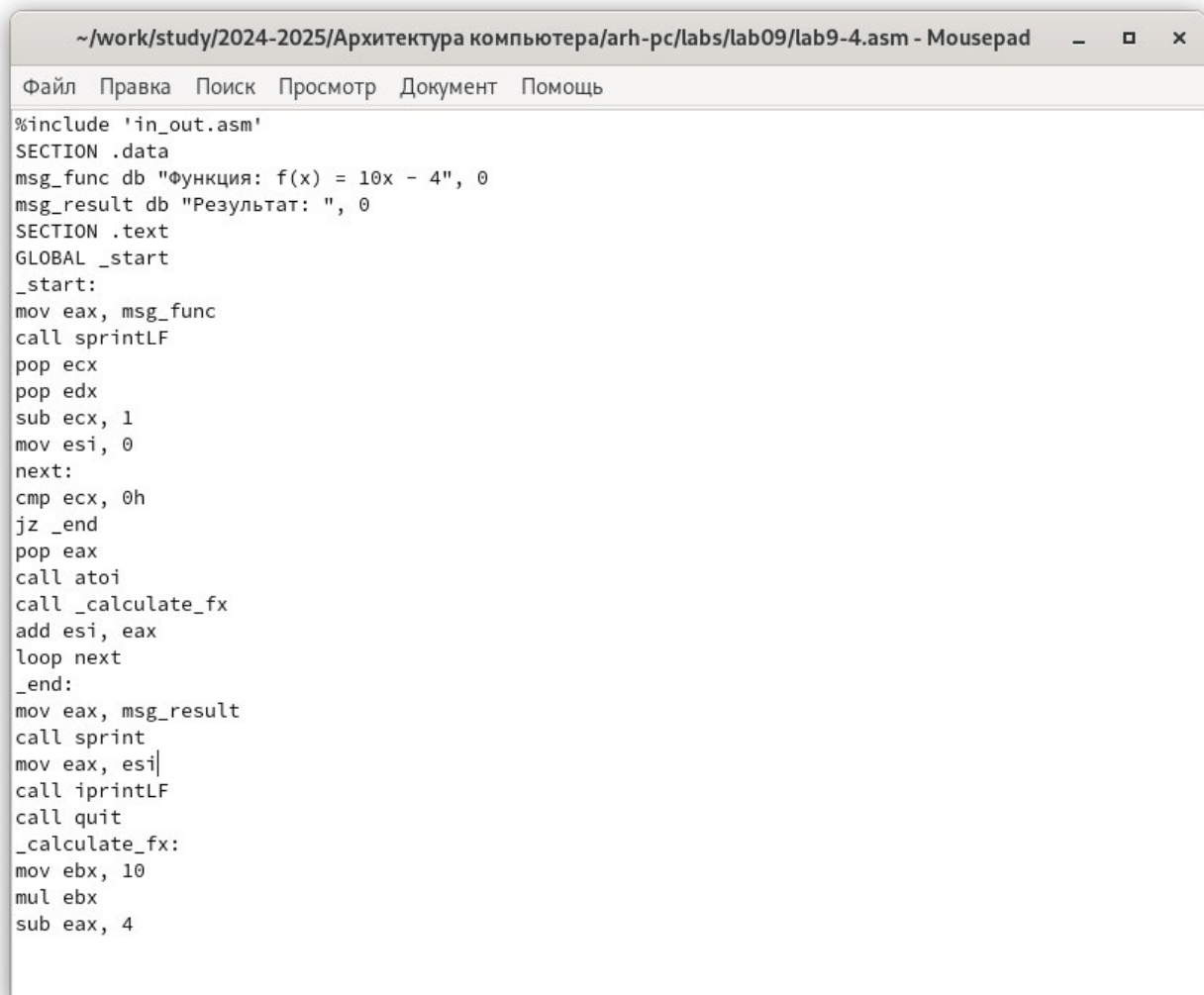
This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество аргументов (первое значение в стеке)
(gdb) x/s *(void**)(esp + 4)
0xffffd152:  "/home/shakira/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd1ba:  "arg1"
(gdb) x/s *(void**)(esp + 12)
0xffffd1bf:  "arg"
(gdb) x/s *(void**)(esp + 16)
0xffffd1c3:  "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd1c5:  "arg 3"
(gdb) x/s *(void**)(esp + 24)
0x0:  <error: Cannot access memory at address 0x0>
(gdb) 
```

Рис.16 Проверка работы стека

## 4.6 Задания для самостоятельной работы

1. Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. 17).



```
~\work\study\2024-2025\Архитектура компьютера\arh-pc\labs\lab09\lab9-4.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

%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
```

Рис.17 Измененная программа предыдущей лабораторной работы

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
```

2. Запускаю программу в режиме отладчика и пошагово через si просматриваю изменение значений регистров через i r. При выполнении инструкции mul ebx 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).



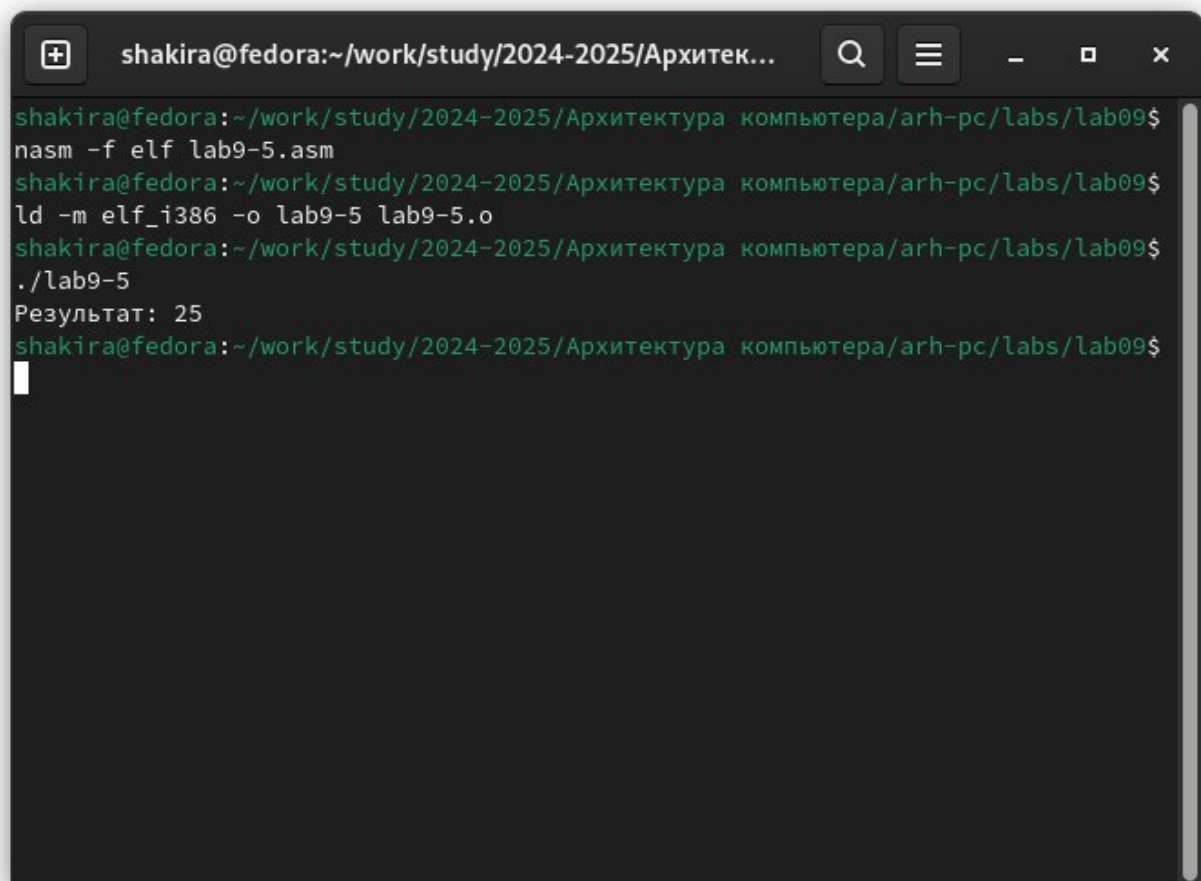
```
shakira@fedora:~/work/study/2024-2025/Архитектура компь...
Register group: general
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490ed 0x80490ed <_start+5>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43

0x80490e0 <quit+5>    mov    $0x1,%eax
0x80490e5 <quit+10>   int     $0x80
0x80490e7 <quit+12>   ret
B+ 0x80490e8 <_start>   mov    $0x804a000,%eax
>0x80490ed <_start+5> call   0x804902d <sprintLF>
0x80490f2 <_start+10> pop     %ecx
0x80490f3 <_start+11> pop     %edx
0x80490f4 <_start+12> sub     $0x1,%ecx

native process 5868 (asm) In: _start L9 PC: 0x80490ed
eax      0x804a000    134520832
ecx      0x0         0
edx      0x0         0
ebx      0x0         0
esp      0xffffcfb0   0xffffcfb0
ebp      0x0         0
esi      0x0         0
edi      0x0         0
eip      0x80490ed    0x80490ed <_start+5>
eflags   0x202       [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис.18 Поиск ошибки в программе через пошаговую отладку

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

A terminal window with a dark background and light green text. The window title is 'shakira@fedora:~/work/study/2024-2025/Архитек...'. The terminal shows the following commands and output:

```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$  
nasm -f elf lab9-5.asm  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$  
ld -m elf_i386 -o lab9-5 lab9-5.o  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$  
./lab9-5  
Результат: 25  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab09$
```

Рис.19 Проверка корректировок в программе

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 помощи GDB и его основными возможностями.

## **6 Источники**

1. [Архитектура ЭВМ \(rudn.ru\)](http://rudn.ru)