

# **Отчет по лабораторной работе №9**

**Дисциплина: архитектура компьютера**

Курушин Георгий Романович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Релаксация подпрограмм в NASM . . . . .	8
4.1.1	Отладка программ с помощью GDB . . . . .	12
4.1.2	Добавление точек останова . . . . .	16
4.1.3	Работа с данными программы в GDB . . . . .	18
4.1.4	Обработка аргументов командной строки в GDB . . . . .	21
4.2	Задание для самостоятельной работы . . . . .	23
<b>5</b>	<b>Выводы</b>	<b>29</b>
<b>6</b>	<b>Список литературы</b>	<b>30</b>

## Список иллюстраций

4.1	Создание рабочего каталога . . . . .	8
4.2	Запуск программы из листинга . . . . .	9
4.3	Изменение программы первого листинга . . . . .	10
4.4	Запуск программы в отладчике . . . . .	12
4.5	Проверка программы отладчиком . . . . .	13
4.6	Запуск отладчика с брейкпойнтом . . . . .	14
4.7	Дисассимилирование программы . . . . .	15
4.8	Режим псевдографики . . . . .	16
4.9	Список брейкпойнтов . . . . .	17
4.10	Добавление второй точки останова . . . . .	17
4.11	Просмотр содержимого регистров . . . . .	18
4.12	Просмотр содержимого переменных двумя способами . . . . .	19
4.13	Изменение содержимого переменных двумя способами . . . . .	19
4.14	Просмотр значения регистра разными представлениями . . . . .	20
4.15	Примеры использования команды set . . . . .	21
4.16	Подготовка новой программы . . . . .	22
4.17	Проверка работы стека . . . . .	23
4.18	Измененная программа предыдущей лабораторной работы . . . . .	24
4.19	Поиск ошибки в программе через пошаговую отладку . . . . .	26
4.20	Проверка корректировок в программе . . . . .	27

## **Список таблиц**

# 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:

- 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, отработывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль).

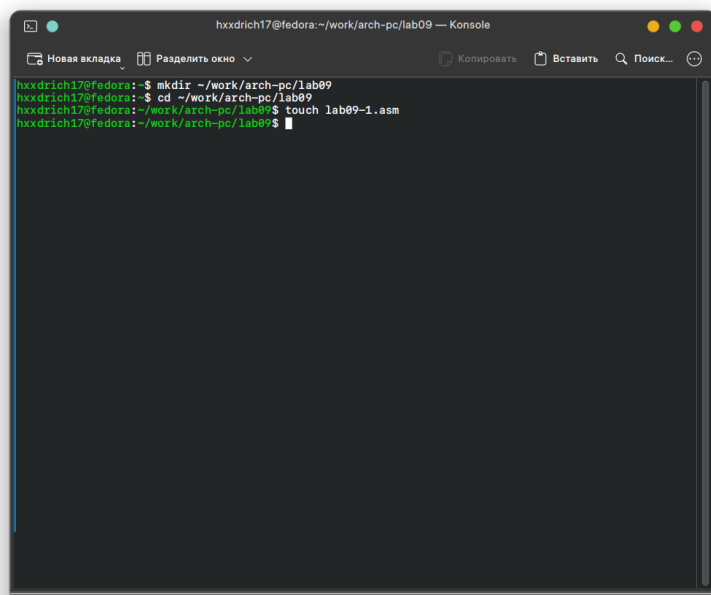
Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга.

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 Релаксация подпрограмм в NASM

Создаю каталог для выполнения лабораторной работы №9 (рис. -fig. 4.1).

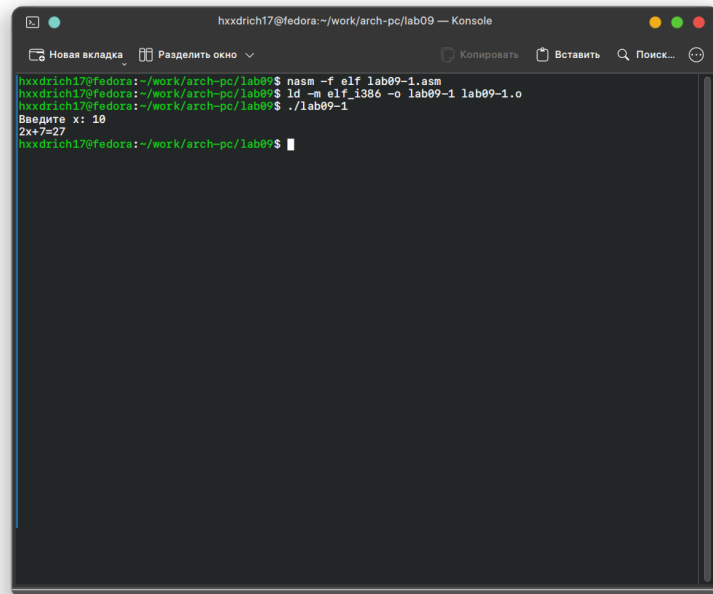


```
hxxdrich17@fedora:~/work/arch-pc/lab09 — Konsole
Новая вкладка Разделить окно Копировать Вставить Поиск...
hxxdrich17@fedora:~$ mkdir ~/work/arch-pc/lab09
hxxdrich17@fedora:~$ cd ~/work/arch-pc/lab09
hxxdrich17@fedora:~/work/arch-pc/lab09$ touch lab09-1.asm
hxxdrich17@fedora:~/work/arch-pc/lab09$
```

Рис. 4.1: Создание рабочего каталога

Копирую в файл код из листинга, компилирую и запускаю его, данная программа выполняет вычисление функции (рис. -fig. 4.2).





```
hxxdrich17@fedora:~/work/arch-pc/lab09 — Konsole
Новая вкладка Разделить окно
hxxdrich17@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
hxxdrich17@fedora:~/work/arch-pc/lab09$ ld -e elf_i386 -o lab09-1 lab09-1.o
hxxdrich17@fedora:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 10
2x+7=27
hxxdrich17@fedora:~/work/arch-pc/lab09$
```

Рис. 4.2: Запуск программы из листинга

Изменяю текст программы, добавив в нее подпрограмму, теперь она вычисляет значение функции для выражения  $f(g(x))$  (рис. -fig. 4.3).

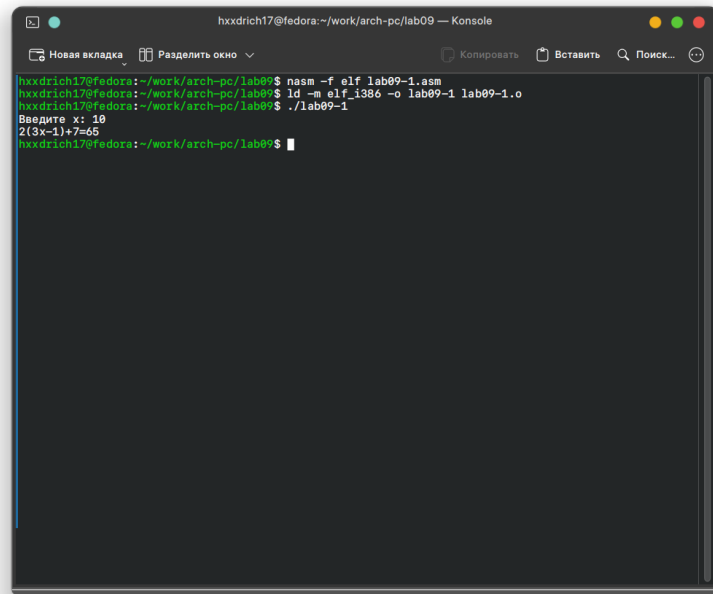


Рис. 4.3: Изменение программы первого листинга

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
```

### 4.1.1 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. -fig. 4.4).

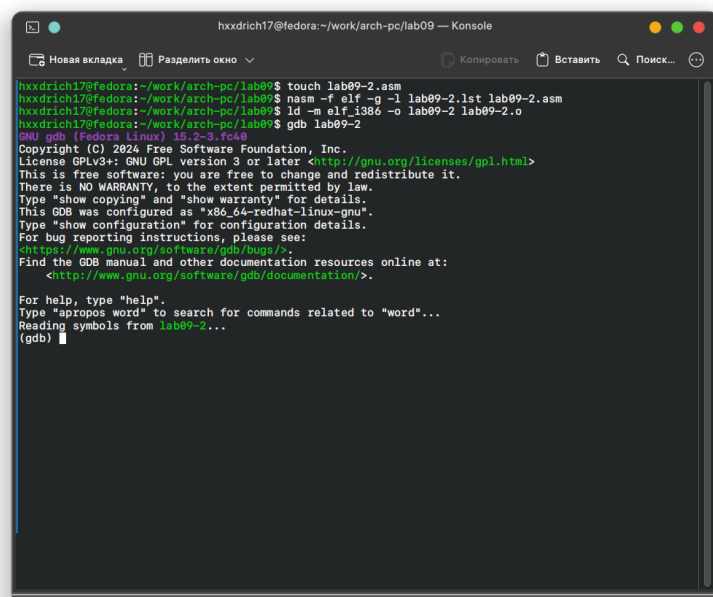
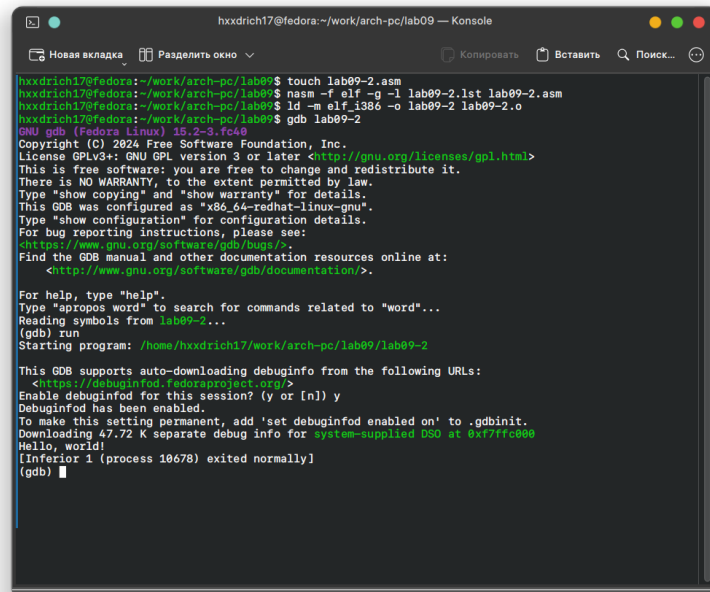


Рис. 4.4: Запуск программы в отладчике

Запустив программу командой `gdb`, я убедился в том, что она работает исправно (рис. -fig. 4.5).

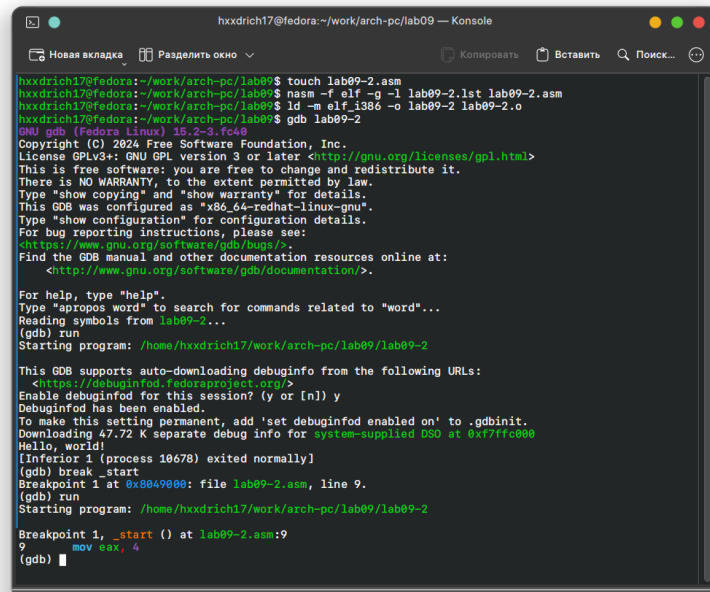


```
hxxdrich17@fedora:~/work/arch-pc/lab09 — Konsole
hxxdrich17@fedora:~/work/arch-pc/lab09$ touch lab09-2.asm
hxxdrich17@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
hxxdrich17@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
hxxdrich17@fedora:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Fedora Linux) 15.2-3.fc48
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 lab09-2...
(gdb) run
Starting program: /home/hxxdrich17/work/arch-pc/lab09/lab09-2
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 47.72 K separate debug info for system-supplied DSO at 0x77fcd000
Hello, world!
Inferior 1 (process 10678) exited normally
(gdb)
```

Рис. 4.5: Проверка программы отладчиком

Для более подробного анализа программы добавляю брейкпоинт на метку `_start` и снова запускаю отладку (рис. -fig. 4.6).



```
hxxdrich17@fedora:~/work/arch-pc/lab09 — Konsole
hxxdrich17@fedora:~/work/arch-pc/lab09$ touch lab09-2.asm
hxxdrich17@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
hxxdrich17@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
hxxdrich17@fedora:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Fedora Linux) 15.2-3.fc48
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 lab09-2...
(gdb) run
Starting program: /home/hxxdrich17/work/arch-pc/lab09/lab09-2
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 47.72 K separate debug info for system-supplied GSO at 0xf7f0000
Hello, world!
(Inferior 1 (process 18678) exited normally)
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/hxxdrich17/work/arch-pc/lab09/lab09-2
Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb) █
```

Рис. 4.6: Запуск отладчика с брейкпоинтом

Далее смотрю дисассимилированный код программы, перевожу на команд с синтаксисом Intel (рис. -fig. 4.7).

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

```
hxxdrich17@fedora:~/work/arch-pc/lab09 — Konsole
[Новая вкладка] [Разделить окно] [Копировать] [Вставить] [Поиск...]
Breakpoint 1 at 0x0049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/hxxdrich17/work/arch-pc/lab09/lab09-2
Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x0049000 <+0>: mov $0x4,%eax
0x0049005 <+5>: mov $0x1,%ebx
0x004900a <+10>: mov $0x04a000,%ecx
0x004900f <+15>: mov $0x8,%edx
0x0049014 <+20>: int $0x80
0x0049016 <+22>: mov $0x4,%eax
0x004901b <+27>: mov $0x1,%ebx
0x0049020 <+32>: mov $0x04a008,%ecx
0x0049025 <+37>: mov $0x7,%edx
0x004902a <+42>: int $0x80
0x004902c <+44>: mov $0x1,%eax
0x0049031 <+49>: mov $0x0,%ebx
0x0049036 <+54>: int $0x80
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x0049000 <+0>: mov eax,0x4
0x0049005 <+5>: mov ebx,0x1
0x004900a <+10>: mov ecx,0x04a000
0x004900f <+15>: mov edx,0x8
0x0049014 <+20>: int 0x80
0x0049016 <+22>: mov eax,0x4
0x004901b <+27>: mov ebx,0x1
0x0049020 <+32>: mov ecx,0x04a008
0x0049025 <+37>: mov edx,0x7
0x004902a <+42>: int 0x80
0x004902c <+44>: mov eax,0x1
0x0049031 <+49>: mov ebx,0x0
0x0049036 <+54>: int 0x80
End of assembler dump.
(gdb) █
```

Рис. 4.7: Дисассимилирование программы

Включаю режим псевдографики для более удобного анализа программы (рис. -fig. 4.8).

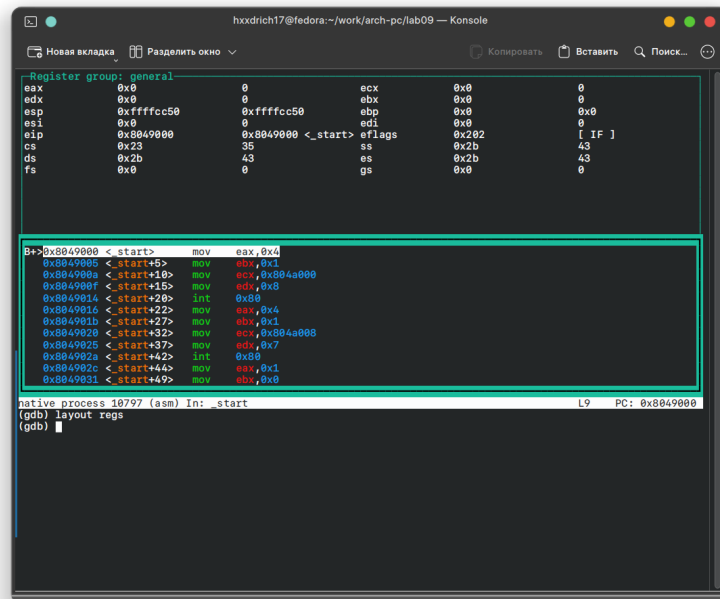


Рис. 4.8: Режим псевдографики

#### 4.1.2 Добавление точек останова

Проверяю в режиме псевдографики, что брейкпоинт сохранился (рис. -fig. 4.9).



```
hxxdrich17@fedora:~/work/arch-pc/lab09 — Konsole
--Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcc60 0      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,0x5
0x8049014 <start+20> int 0x80
0x8049016 <start+22> mov eax,0x4
0x804901b <start+27> mov ebx,0x1
0x8049020 <start+32> mov ecx,0x804a000
0x8049025 <start+37> mov edx,0x7
0x804902a <start+42> int 0x80
0x804902c <start+44> mov ebx,0x1
b> 0x8049031 <start+49> mov ebx,0x0

native process 11501 (asm) In: start L9 PC: 0x8049000
(gdb) layout regs
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb) i b
Num Type Disp Enb Address What
1 breakpoint keep y 0x8049000 lab09-2.asm:9
2 breakpoint already hit 1 time
breakpoint keep y 0x8049031 lab09-2.asm:20
(gdb) |
```

Рис. 4.9: Список брейкпоинтов

Устанавливаю еще одну точку останова по адресу инструкции (рис. -fig. 4.10).

```
hxxdrich17@fedora:~/work/arch-pc/lab09 — Konsole
--Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcc60 0      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

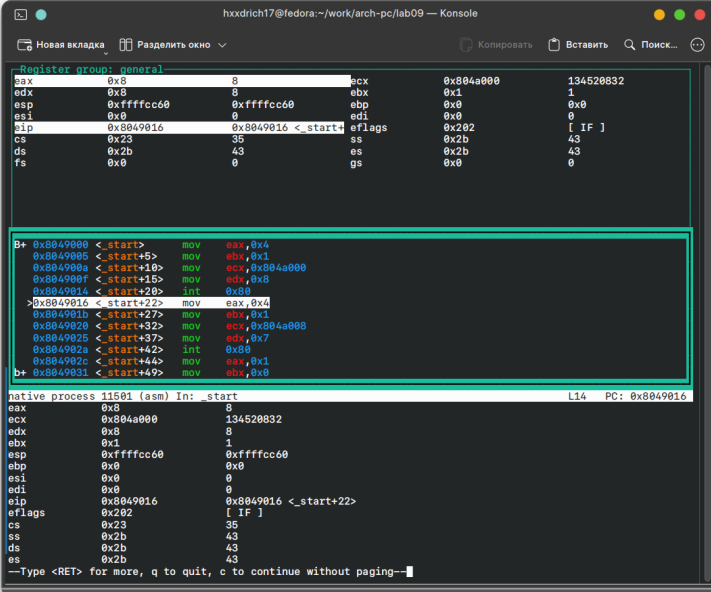
B> 0x8049000 <start> mov eax,0x4
0x8049005 <start+5> mov ebx,0x1
0x804900a <start+10> mov ecx,0x804a000
0x804900f <start+15> mov edx,0x5
0x8049014 <start+20> int 0x80
0x8049016 <start+22> mov eax,0x4
0x804901b <start+27> mov ebx,0x1
0x8049020 <start+32> mov ecx,0x804a000
0x8049025 <start+37> mov edx,0x7
0x804902a <start+42> int 0x80
0x804902c <start+44> mov ebx,0x1
b> 0x8049031 <start+49> mov ebx,0x0

native process 11501 (asm) In: start L9 PC: 0x8049000
(gdb) layout regs
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb) i b
Num Type Disp Enb Address What
1 breakpoint keep y 0x8049000 lab09-2.asm:9
2 breakpoint already hit 1 time
breakpoint keep y 0x8049031 lab09-2.asm:20
(gdb) |
```

Рис. 4.10: Добавление второй точки останова

### 4.1.3 Работа с данными программы в GDB

Просматриваю содержимое регистров командой `info registers` (рис. -fig. 4.11).



```
hxdrich17@fedora:~/work/arch-pc/lab09 — Konsole
[New Window] [Split Window] [Copy] [Paste] [Search]
Register group: general
eax      0x8      8      ecx      0x804a000      134520832
edx      0x0      0      ebx      0x1      1
esp      0xffffcc60 0xffffcc60  ebp      0x0      0
esi      0x0      0      edi      0x0      0
eip      0x8049016 0x8049016 <_start+  eflags    0x202      [ IF ]
ds       0x2b      43      ss       0x2b      43
fs       0x0      0      gs       0x0      0

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,0x804a000
0x8049025 <_start+37> mov     edx,0x7
0x804902a <_start+42> int     0x80
0x804902c <_start+44> mov     esi,0x1
0x8049031 <_start+49> mov     ebx,0x0

native process 11501 (asm) In: _start      L14  PC: 0x8049016
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffcc60 0xffffcc60
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags    0x202      [ IF ]
ds       0x2b      35
ss       0x2b      43
ds       0x2b      43
es       0x2b      43
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис. 4.11: Просмотр содержимого регистров

Смотрю содержимое переменных по имени и по адресу (рис. -fig. 4.12).

```

hxxdrich17@fedora:~/work/arch-pc/lab09 — Konsole

--Register group: general
eax 0x8 8 ecx 0x804a000 134528832
edx 0x0 0 ebx 0x1 1
esp 0xffffcc60 0xffffcc60 ebp 0x0 0x0
esi 0x0 0 edi 0x0 0
eip 0x8049016 0x8049016 <_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
0x804902c <_start+44> mov eax,0x1
b* 0x8049031 <_start+49> mov ebx,0x0

native process 11501 (asm) In: _start L14 PC: 0x8049016
eax 0x0 0 ebx 0x1 1
eip 0x8049016 0x8049016 <_start+22>
eflags 0x202 [ IF ]
cs 0x23 35
ss 0x2b 43
ds 0x2b 43
es 0x2b 43
fs 0x0 0
gs 0x0 0
--Type <RET> for more, q to quit, c to continue without paging--
(gdb) x/1sb &msg1
0x8049000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb)

```

Рис. 4.12: Просмотр содержимого переменных двумя способами

Меняю содержимое переменных по имени и по адресу (рис. -fig. 4.13).

```

hxxdrich17@fedora:~/work/arch-pc/lab09 — Konsole

--Register group: general
eax 0x8 8 ecx 0x804a000 134528832
edx 0x0 0 ebx 0x1 1
esp 0xffffcc60 0xffffcc60 ebp 0x0 0x0
esi 0x0 0 edi 0x0 0
eip 0x8049016 0x8049016 <_start+ eflags 0x202 [ IF ]
cs 0x23 35 ss 0x2b 43
ds 0x2b 43 es 0x2b 43
fs 0x0 0 gs 0x0 0

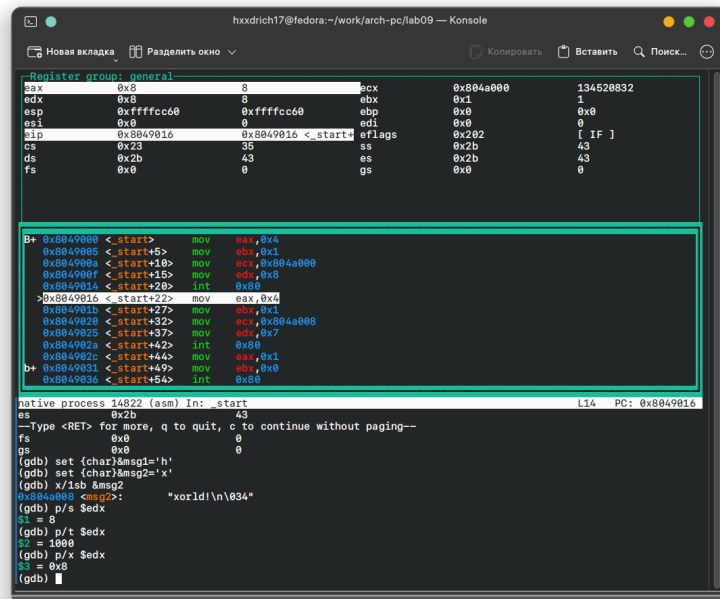
B* 0x8049000 <_start> mov eax,0x4
0x8049005 <_start+5> mov ebx,0x1
0x804900a <_start+10> mov ecx,0x804a000
0x804900f <_start+15> mov edx,0x8
0x8049014 <_start+20> int 0x80
>0x8049016 <_start+22> mov eax,0x4
0x804901b <_start+27> mov ebx,0x1
0x8049020 <_start+32> mov ecx,0x804a008
0x8049025 <_start+37> mov edx,0x7
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1
b* 0x8049031 <_start+49> mov ebx,0x0

native process 11501 (asm) In: _start L14 PC: 0x8049016
eax 0x0 0 ebx 0x1 1
eip 0x8049016 0x8049016 <_start+22>
eflags 0x202 [ IF ]
cs 0x23 35
ss 0x2b 43
ds 0x2b 43
es 0x2b 43
fs 0x0 0
gs 0x0 0
--Type <RET> for more, q to quit, c to continue without paging--
(gdb) x/1sb &msg1
0x8049000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb) set {char}&msg1="h"
(gdb) x/1sb &msg1
0x8049000 <msg1>: "hello, "
(gdb) set {char}&msg2="x"
(gdb) x/1sb &msg2
0x804a008 <msg2>: "xor!\n\034"
(gdb)

```

Рис. 4.13: Изменение содержимого переменных двумя способами

Вывожу в различных форматах значение регистра edx (рис. -fig. 4.14).



The screenshot shows a GDB console window with the following content:

```
Register group: general
eax 0x8 8 ecx 0x804a000 134520832
edx 0x8 8 ebx 0x1 1
esp 0xffffcc60 0xffffcc60 ebp 0x0 0x0
esi 0x0 0x0 edi 0x0 0x0
eip 0x8049016 0x8049016 <start+... eflags 0x202 [ IF ]
cs 0x23 43 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,0x0
0x8049014 <start+20> int 0x80
>0x8049016 <start+22> mov eax,0x4
0x804901b <start+27> mov ebx,0x1
0x8049020 <start+32> mov ecx,0x804a000
0x8049025 <start+37> mov edx,0x7
0x804902a <start+42> int 0x80
0x804902c <start+44> mov esi,0x1
0x8049031 <start+49> mov edi,0x0
B* 0x8049036 <start+54> int 0x80
```

```
native process 14822 (asm) In: start L14 PC: 0x8049016
es 0x2b 43
--Type <RET> for more, q to quit, c to continue without paging--
fs 0x0 0
gs 0x0 0
(gdb) set {char}&msg1='h'
(gdb) set {char}&msg2='x'
(gdb) x/1sb &msg2
0x8049035 <msg2>: "xorld!\n\034"
(gdb) p/s $edx
$1 = 8
(gdb) p/t $edx
$2 = 1000
(gdb) p/x $edx
$3 = 0x8
(gdb)
```

Рис. 4.14: Просмотр значения регистра разными представлениями

С помощью команды set меняю содержимое регистра ebx (рис. -fig. 4.15).

```

hxxdrich17@fedora: ~/work/arch-pc/lab09 — Konsole
--Register group: general--
eax      0x8          8          ecx      0x804a000      134520832
edx      0x0          0          ebx      0x2          2
esp      0xffffcc60   0xffffcc60   ebp      0x0          0x0
esi      0x0          0          edi      0x0          0
eip      0x8049016    0x8049016    eflags   0x202         [ IF ]
cs       0x23         35          ss       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    edi,0x8
0x8049014 <start+20> int    0x80
>0x8049016 <start+22> mov    eax,0x4
0x804901b <start+27> mov    ebx,0x1
0x8049020 <start+32> mov    ecx,0x804a000
0x8049025 <start+37> mov    edi,0x7
0x804902a <start+42> int    0x80
0x804902c <start+44> mov    eax,0x1
B> 0x8049031 <start+49> mov    ebx,0x0
0x8049036 <start+54> int    0x80

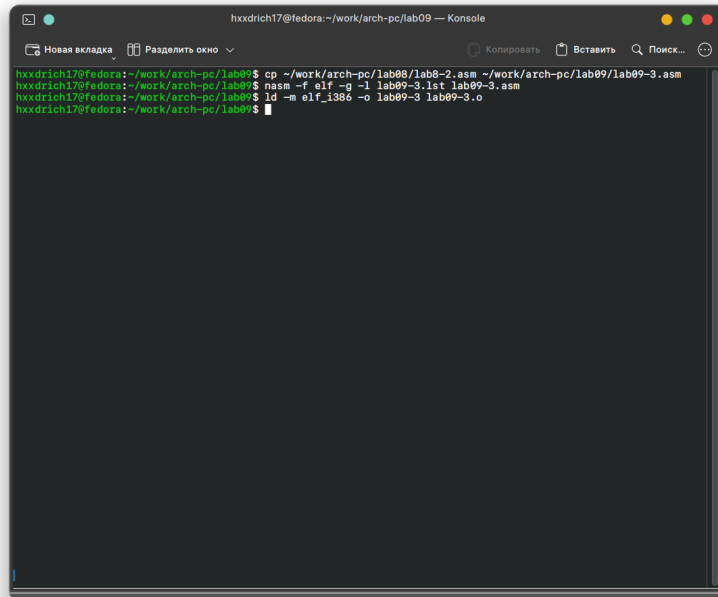
native process 14822 (asm) In: start L14 PC: 0x8049016
(gdb) p/s $edx
$1 = 8
(gdb) p/t $edx
$2 = 1000
(gdb) p/x $edx
$3 = 0x8
(gdb) set $ebx='Z'
(gdb) p/s
$4 = 8
(gdb) p/s $ebx
$5 = 80
(gdb) sat $ebx=2
(gdb) p/s $ebx
$6 = 2
(gdb)

```

Рис. 4.15: Примеры использования команды set

#### 4.1.4 Обработка аргументов командной строки в GDB

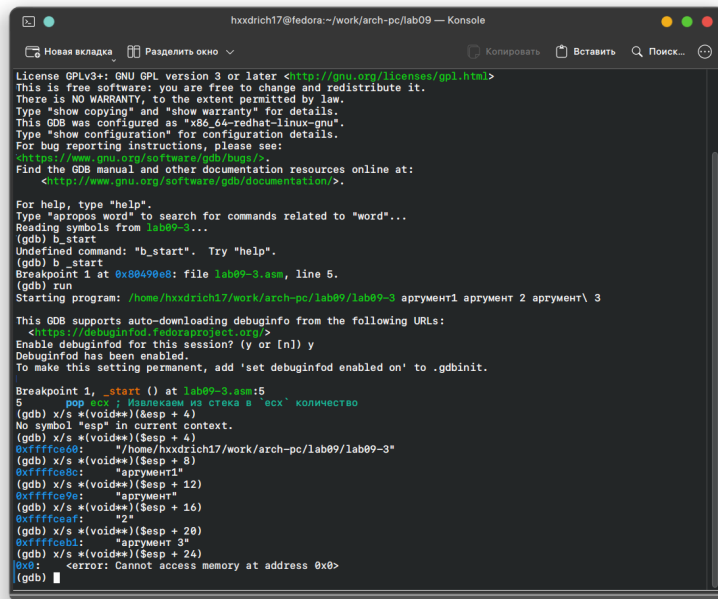
Копирую программу из предыдущей лабораторной работы в текущий каталог и и создаю исполняемый файл с файлом листинга и отладки (рис. -fig. 4.16).



```
hxxdrich17@fedora:~/work/arch-pc/lab09 — Konsole
Новая вкладка  Разделить окно  Копировать  Вставить  Поиск...
hxxdrich17@fedora:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
hxxdrich17@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
hxxdrich17@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
hxxdrich17@fedora:~/work/arch-pc/lab09$
```

Рис. 4.16: 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. -fig. 4.17).



```
hxxdrich17@fedora:~/work/arch-pc/lab09 — Konsole
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:
  <http://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 lab09-3...
(gdb) b _start
Undefined command: "b_start". Try "help".
(gdb) b _start
Breakpoint 1 at 0x8049000: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/hxxdrich17/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 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 lab09-3.asm:5
5      pop     ecx; количество элементов из стека в 'ecx' количество
(gdb) x/s *(void**)(esp + 4)
0xffffc000: /home/hxxdrich17/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffc000: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffc000: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffc000: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffc000: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 4.17: Проверка работы стека

## 4.2 Задание для самостоятельной работы

1. Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. -fig. 4.18).

```
lab09-4.asm
~/Рабочий стол/study labs/lab09

lab8-3.asm | lab8-4.asm | lab09-1.asm | lab09-2.asm | lab09-4.asm x

#include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 30x - 11", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
mov eax, msg_func
call sprintf

pop ecx
pop edx
sub ecx, 1
mov esi, 0

next:
cmp ecx, 0h
jz _end
pop eax
call atoi

call _calculate_func

add esi, eax

loop next

_end:
mov eax, msg_result
call sprintf
mov eax, esi
call iprintfLF
call quit

_calculate_func:
mov ebx, 30
mul ebx
sub eax, 11
ret
```

Рис. 4.18: Измененная программа предыдущей лабораторной работы

Код программы:

```
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 30x - 11", 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_func

add esi, eax

loop next

_end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit

```

```

_calculate_func:
mov ebx, 30
mul ebx
sub eax, 11
ret

```

2. Запускаю программу в режиме отладчика и пошагово через si просматриваю изменение значений регистров через i r. При выполнении инструкции mul esx 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. -fig. 4.19).

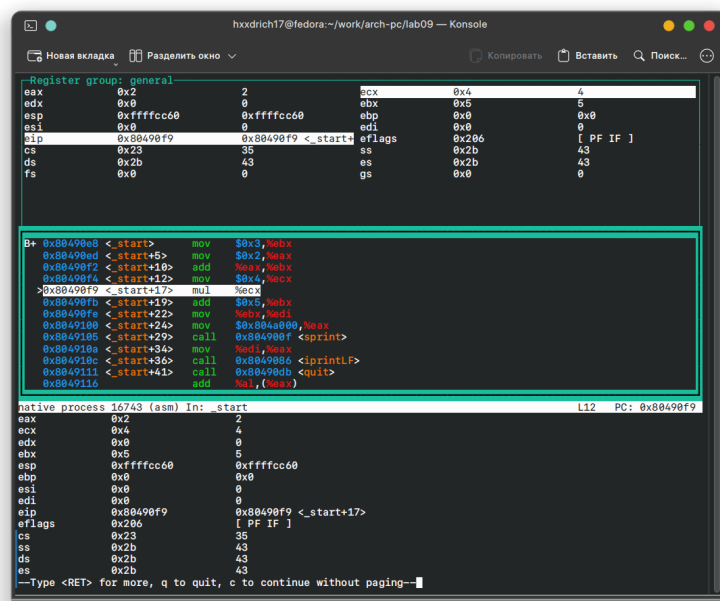


Рис. 4.19: Поиск ошибки в программе через пошаговую отладку

Исправляю найденную ошибку, теперь программа верно считает значение функции (рис. -fig. 4.20).

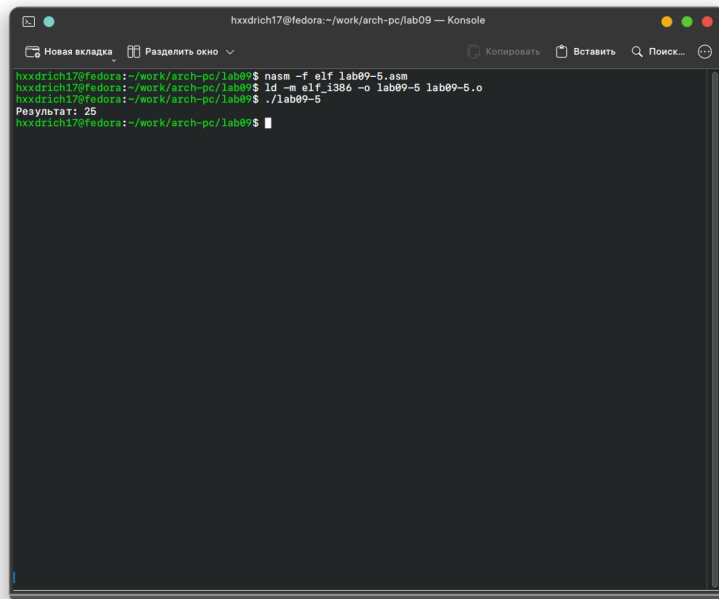


Рис. 4.20: Проверка корректировок в программе

Код измененной 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 приобрел навыки написания программ с использованием подпрограмм, а так же познакомился с методами отладки при помощи GDB и его основными возможностями.

## **6 Список литературы**

1. Курс на ТУИС
2. Лабораторная работа №9
3. Программирование на языке ассемблера NASM Столяров А. В.