

# Отчет по лабораторной работе №9

## Дисциплина: архитектура компьютера

Гаязов Рузаль Ильшатovich

### Содержание

Цель работы.....	1
Теоретическое введение.....	1
Выполнение лабораторной работы.....	2
Задание для самостоятельной работы.....	7
Выводы.....	10
Список литературы.....	10

### Цель работы

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

### 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 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.

## Выполнение лабораторной работы

Создаю файл lab9-1.asm (рис. -@fig:001).

```
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09
mc [ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьют... x ruzalgayazov@fedora:~/work/study/2024-2025/Ar
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cd labs
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs$ cd lab09
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ls
presentation report
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ touch lab9-1.asm
```

### Создание файла

Копирую в файл код из листинга, компилирую и запускаю его (рис. -@fig:002).

```
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ nasm -f elf lab9-1.asm
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ./lab9-1
Введите x: 10
2x7=27
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$
```

### Запуск программы

Изменяю текст программы, добавив в нее подпрограмму (рис. -@fig:003).

```
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ nasm -f elf lab9-1.asm
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ./lab9-1
Введите x: 10
2(3x-1)+7=65
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$
```

### Изменение программы

В созданный файл копирую программу второго листинга, транслирую с созданием файла листинга и отладки, компоную и запускаю в отладчике. Запустил программу командой run (рис. -@fig:004).

```
ld: cannot find lab9-2.o: No such file or directory
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 15.2-2.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) run
Starting program: /home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-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 47.71 K separate debug info for system-supplied DSO at 0xfffffc000
Hello, world!
[Inferior 1 (process 2908) exited normally]
(gdb)
```

### Запуск программы

Для более подробного анализа программы добавляю брейкпоинт на метку \_start и снова запускаю отладку (рис. -@fig:005).

```

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/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-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 47.71 K separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 2908) exited normally]
(gdb) break _start
Breakpoint 1 at 0x049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09/lab9-2

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

```

## Запуск отладчика

Далее смотрю дисассимилированный код программы, перевожу на команд с синтаксисом Intel (рис. -@fig:006). 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 ax, eax, непосредственные операнды пишутся напрямую), именах регистров (АТТ - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

```

(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     $0x804a000,%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,0x804a000
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)

```

## Дисассимилирование программы

Включаю режим псевдографики для более удобного анализа программы (рис. -@fig:007).

```
rusalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09 — gdb lab9-2
mc [rusalgayazov@fedora:~/work/study/2024-2025/Архитектура компьют... x rusalgayazov@fedora:~/work/study/2024-2025/Архитектура
Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcf10 0xffffcf10  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,0
0x804900a <_start+10> mov     ecx,0x8049000
0x804900f <_start+15> mov     edx,0x8
0x8049014 <_start+20> int     0x80
0x8049019 <_start+25> mov     eax,0x4
0x804901e <_start+30> mov     ebx,0x1
0x8049023 <_start+35> mov     ecx,0x8049000
0x8049028 <_start+40> mov     edx,0x7
0x804902d <_start+45> int     0x80

native process 2929 (asm) In: _start
(gdb) layout regs
(gdb)
```

## Режим псевдографики

Проверяю в режиме псевдографики, что брейкпоинт сохранился (рис. -@fig:008).

```
(gdb) i b
Num      Type      Disp Enb Address      What
1        breakpoint keep y  0x08049000 lab9-2.asm:9
breakpoint already hit 1 time
(gdb)
```

## Список брейкпоинтов

Устанавливаю еще одну точку останова по адресу инструкции (рис. -@fig:009).

```
native process 2929 (asm) In: _start
Num      Type      Disp Enb Address      What
1        breakpoint keep y  0x08049000 lab9-2.asm:9
breakpoint already hit 1 time
(gdb) b *0x08049031
Breakpoint 2 at 0x08049031: file lab9-2.asm, line 20.
(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  0x08049031 lab9-2.asm:20
(gdb)
```

## Добавление точки останова

Просматриваю содержимое регистров командой info registers (рис. -@fig:010).

```
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09 — gdb lab9
mc [ruzalgayazov@fedora]:~/work/study/2024-2025/Архитектура компьют... x ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьют...

--Register group: general--
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcf10 0xffffcf10  ebp      0x0      0
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

0x8049310 add BYTE PTR [eax],%1
0x804931a add BYTE PTR [eax],%1
0x804931c add BYTE PTR [eax],%1
0x804931e add BYTE PTR [eax],%1
0x8049320 add BYTE PTR [eax],%1
0x8049322 add BYTE PTR [eax],%1
0x8049324 add BYTE PTR [eax],%1
0x8049326 add BYTE PTR [eax],%1
0x8049328 add BYTE PTR [eax],%1
0x804932a add BYTE PTR [eax],%1

native process 2929 (asm) In: _start
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcf10 0xffffcf10
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--
```

Просмотр содержимого регистров

Смотрю содержимое переменных по имени и по адресу (рис. -@fig:011).

```
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09 — gdb lab9-2
mc [ruzalgayazov@fedora]:~/work/study/2024-2025/Apx... x ruzalgayazov@fedora:~/work/study/2024-2025/Apx...

--Register group: general--
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcf10 0xffffcf10  ebp      0x0      0
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

0x8049400 add BYTE PTR [eax],%1
0x8049402 add BYTE PTR [eax],%1
0x8049404 add BYTE PTR [eax],%1
0x8049406 add BYTE PTR [eax],%1
0x8049408 add BYTE PTR [eax],%1
0x804940a add BYTE PTR [eax],%1
0x804940c add BYTE PTR [eax],%1
0x804940e add BYTE PTR [eax],%1
0x8049410 add BYTE PTR [eax],%1

native process 2929 (asm) In: _start
L9
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) x/1sb &msg1
0x8049400 <msg1>:      "Hello, "
(gdb) s
```

Просмотр содержимого

Меняю содержимое переменных по имени и по адресу (рис. -@fig:012).

```
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09 — gdb lab9-2
mc [ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09] x ruzalgayazov@fedora:~/work/study/2024-2025/Архитек... x
--Register group: general--
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcf10 0xffffcf10  ebp      0x0      0
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

0x8049000 add BYTE PTR [eax],1
0x8049001 add BYTE PTR [eax],1
0x8049002 add BYTE PTR [eax],1
0x8049003 add BYTE PTR [eax],1
0x8049004 add BYTE PTR [eax],1
0x8049005 add BYTE PTR [eax],1
0x8049006 add BYTE PTR [eax],1
0x8049007 add BYTE PTR [eax],1
0x8049008 add BYTE PTR [eax],1
0x8049009 add BYTE PTR [eax],1
0x804900a add BYTE PTR [eax],1
0x804900b add BYTE PTR [eax],1
0x804900c add BYTE PTR [eax],1
0x804900d add BYTE PTR [eax],1
0x804900e add BYTE PTR [eax],1
0x804900f add BYTE PTR [eax],1
0x8049010 add BYTE PTR [eax],1

native process 2929 (asm) In: _start L9 PC: 0x8049000
(gdb) set {char}msg1='Hello, "
'msg1' has unknown type; cast it to its declared type
(gdb) x/1sb &msg2
0x8049000 <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
0x8049000 <msg2>: "xorld!\n\034"
(gdb)
```

## Изменение содержимого

Вывожу в различных форматах значение регистра edx (рис. -@fig:013).

```
--Register group: general--
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd070 0xffffd070
ebp      0x0      0x0
esi      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

native process 10469 (asm) In: _start L15 PC: 0x8049016
(gdb) p/t $ecx
$2 = 10000000010010100000000000000000
(gdb) p/s $edx
$3 = 8
(gdb) p/t $edx
$4 = 1000
(gdb) p/x $edx
$5 = 0x8
(gdb)
```

## Просмотр значения регистра разными представлениями

С помощью команды set меняю содержимое регистра ebx (рис. -@fig:014).



```

--Register group: general--
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x2      2
esp      0xffffd070 0xffffd070
ebp      0x0      0x0
esi      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

native process 10469 (asm) In: _start      L15      PC: 0x8049016
(gdb) set $ebx='2'
(gdb) p/s
$6 = 8
(gdb) p/s $ebx
$7 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$8 = 2
(gdb)

```

### Примеры использования set

Копирую программу из предыдущей лабораторной работы в текущий каталог. 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:015).

```

(gdb) x/s *(void**)(esp + 4)
0xfffff010: 0x01
(gdb) x/s *(void**)(esp + 8)
0xfffff010: "/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 12)
0xfffff010: <error: Cannot access memory at address 0x0>
(gdb) x/s *(void**)(esp + 16)
0xfffff010: "SHELL=/bin/bash"
(gdb) x/s *(void**)(esp + 20)
0xfffff010: "SESSION_MANAGER=local/unix:@/tmp/.ICE-unix/1283,unix:/tmp/.ICE-unix/1283"
(gdb) x/s *(void**)(esp + 24)
0xfffff010: "COLORTERM=truecolor"
(gdb) x/s *(void**)(esp + 28)
0xfffff010: "HISTCONTROL=ignoredups"
(gdb)

```

### Проверка работы стека

## Задание для самостоятельной работы

Меняю программу самостоятельной части с использованием подпрограммы (рис. -@fig:016).

```
mc [ruzalgayazov@fedora]:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09
/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09/lab9-4.asm
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x)=4x+3", 10, 0
msg db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
    mov eax, msg_func
    call sprint
    pop ecx
    sub ecx, 1
    mov esi, 0

next:
    cmp ecx, 0
    jz _end
    pop eax
    call atoi
    push eax
    call calculate_function
    add esi, eax
    dec ecx
    jmp next

_end:
    mov eax, msg
    call sprint
    mov eax, esi
    call iprintfLF
    call quit

calculate_function:
    pop eax
    lea eax, [eax*4+3]
    ret
```

*Измененная программа*

Код программы:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg_func db "Функция: f(x)=4x+3", 10, 0
```

```
msg db "Результат: ", 0
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
    mov eax, msg_func
```

```
    call sprint
```

```
    pop ecx
```

```
    sub ecx, 1
```

```
    mov esi, 0
```

```
next:
```

```
    cmp ecx, 0
```

```
    jz _end
```

```
    pop eax
```

```
    call atoi
```

```
    push eax
```

```
    call calculate_function
```

```
    add esi, eax
```

```
    dec ecx
```

```
    jmp next
```



\_end:

```
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

calculate\_function:

```
pop eax
lea eax, [eax*4+3]
ret
```

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:017).

```
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09 — gdb lab9-4
mc [ruzalgayazov@fedora:~/wor... x ruzalgayazov@fedora:~/work/stu... x ruzalgayazov@fedora:~/work/stu... x
Register group: general
eax 0x0 0 ecx 0x0 0
edx 0x0 0 ebx 0x0 0
esp 0xffffcf70 0xffffcf70 ebp 0x0 0x0
esi 0x0 0 edi 0x0 0
eip 0x80490e8 0x80490e8 <_start> eflags 0x202 [ IF ]
cs 0x23 35 ss 0x2b 43
ds 0x2b 43 es 0x2b 43
fs 0x0 0 gs 0x0 0

B>0x80490e8 <_start> mov $0x804a000,%eax
0x80490ed <_start+5> call 0x80490e8 <sprint>
0x80490f2 <_start+10> pop %ecx
0x80490f3 <_start+11> sub $0x1,%ecx
0x80490f6 <_start+14> mov $0x0,%esi
0x80490fb <next> cmp $0x0,%ecx
0x80490fe <next+3> je 0x8049111 <_end>
0x8049100 <next+5> pop %eax
0x8049101 <next+6> call 0x804909c <atoi>
0x8049106 <next+11> push %eax
0x8049107 <next+12> call 0x8049127 <calculate_function>
0x804910c <next+17> add %eax,%esi

native process 3681 (asm) in: _start L11 PC: 0x80490e8
(gdb) layout regs
(gdb) r 1
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09/lab9-4 1
Breakpoint 1, _start () at lab9-4.asm:11
(gdb) r 1
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09/lab9-4 1
Breakpoint 1, _start () at lab9-4.asm:11
(gdb)
```

## Поиск ошибки

Исправляю найденную ошибку, теперь программа верно считает значение функции (рис. -@fig:018).

```
mc [ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09] Q x
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ nasm -f elf -g -l lab9-4.lst lab9-4.asm
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o lab9-4 lab9-4.o
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ./lab9-4
bash: ./lab9-4.asm: Permission denied
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o lab9-4 lab9-4.o
lab9-4.asm: file not recognized: file format not recognized
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ nasm -f elf lab5-1.asm
nasm: fatal: unable to open input file 'lab5-1.asm' No such file or directory
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ nasm -f elf lab9-4.asm
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o lab9-4 lab9-4.o
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ./lab9-4
Результат: 25
ruzalgayazov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$
```

## 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
```

## 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.

## Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.

3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс,
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: [http://www.stolyarov.info/books/asm\\_unix](http://www.stolyarov.info/books/asm_unix).
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).