

Отчёт по лабораторной работе №9

Понятие подпрограммы. Отладчик.

Исупов Олег Денисович

Содержание

1	Цель работы	6
2	Выполнение лабораторной работы	7
3	Задания для самостоятельной работы	18
4	Выводы	22

Список иллюстраций

2.1	Создание файла	7
2.2	Заполнение файла	8
2.3	Проверка	8
2.4	Редактирование файла	9
2.5	Проверка	10
2.6	Создание файла	10
2.7	Заполнение файла	10
2.8	Загрузка исходного файла в отладчик	11
2.9	Запуск команды	11
2.10	Запуск программы с брейкпоинтом	11
2.11	Просмотр дисассимилированного кода программы	12
2.12	Переключение на синтаксис Intel	12
2.13	Включение отображения регистров, их значений и результата программы	13
2.14	Использование команды info breakpoints	13
2.15	Создание новой точки	14
2.16	Просмотр информации	14
2.17	Отслеживание регистров	15
2.18	Просмотр значения переменной	15
2.19	Просмотр значения переменной	15
2.20	Изменение символа	15
2.21	Просмотр значения переменной	16
2.22	Просмотр значения регистра	16
2.23	Изменение регистра командой set	16
2.24	Прописывание команд с и quit	16
2.25	Копирование файла	16
2.26	Создание и запуск файл в отладчике	17
2.27	Устанавливаем точку	17
2.28	Изучение полученных данных	17
3.1	Копирование и изменение файла	18
3.2	Проверка	18
3.3	Создание файла	19
3.4	Изменение файла	19
3.5	Проверка	19
3.6	Поиск ошибки регистров в отладчике	20
3.7	Изменение файла	20

3.8 Проверка	21
------------------------	----

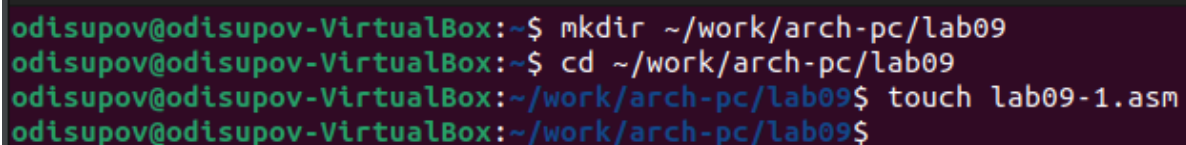
Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

1. Создайте каталог для выполнения лабораторной работы № 9, перейдите в него и создайте файл lab09-1.asm



```
odisupov@odisupov-VirtualBox:~$ mkdir ~/work/arch-pc/lab09
odisupov@odisupov-VirtualBox:~$ cd ~/work/arch-pc/lab09
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ touch lab09-1.asm
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$
```

Рис. 2.1: Создание файла

2. В качестве примера рассмотрим программу вычисления арифметического выражения $x(x) = 2x + 7$ с помощью подпрограммы _calcul. В данном примере x вводится с клавиатуры, а само выражение вычисляется в подпрограмме.

```

GNU nano 6.2 /home/odisupov/work/arch-pc/lab09/lab09-1.asm *
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
res: RESB 80
SECTION .text
GLOBAL _start
_start:
;-----
; Основная программа
;-----
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi

```

Рис. 2.2: Заполнение файла

```

odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab
09-1.o
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 5
2x+7=17
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$

```

Рис. 2.3: Проверка


```

GNU nano 6.2
#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:
;-----
; Основная программа
;-----
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax,x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax,result
call sprint
mov eax,[res]
call iprintLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
call _subcalcul
mov ebx,2
mul ebx
add eax,7
mov [res],eax
ret ; выход из подпрограммы
_subcalcul:
mov ebx,3
mul ebx
sub eax,1
ret

```

Рис. 2.4: Редактирование файла

```

odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 5
2(3x-1)+7=35
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$

```

Рис. 2.5: Проверка

3. Создайте файл lab09-2.asm с текстом программы из Листинга 9.2

```

odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ touch lab09-2.asm
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$

```

Рис. 2.6: Создание файла

```

GNU nano 6.2 /home/odisupov/work/arch-pc/lab09/lab09-2.asm *
SECTION .data
msg1: db "Hello, ",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
msg2Len: equ $ - msg2
SECTION .text
global _start
_start:
mov eax, 4
mov ebx, 1
mov ecx, msg1
mov edx, msg1Len
int 0x80
mov eax, 4
mov ebx, 1
mov ecx, msg2
mov edx, msg2Len
int 0x80
mov eax, 1
mov ebx, 0

```

Рис. 2.7: Заполнение файла

```

odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst
lab09-2.asm
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab
09-2.o
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 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-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"...

```

Рис. 2.8: Загрузка исходного файла в отладчик

```

(gdb) run
Starting program: /home/odisupov/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 96600) exited normally]
(gdb)

```

Рис. 2.9: Запуск команды

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/odisupov/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb)

```

Рис. 2.10: Запуск программы с брейкпоинтом

```

(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)

```

Рис. 2.11: Просмотр дисассимилированного кода программы

```

(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. 2.12: Переключение на синтаксис Intel

```

[ Register Values Unavailable ]

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 96620 In: _start          L9      PC: 0x8049000
(gdb) layout regs
(gdb)

```

Рис. 2.13: Включение отображения регистров, их значений и результата программы

```

[ Register Values Unavailable ]

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 96620 In: _start          L9      PC: 0x8049000
(gdb) layout regs
(gdb) info breakpoints
Num      Type             Disp Enb Address          What
1        breakpoint       keep y   0x08049000 lab09-2.asm:9
breakpoint already hit 1 time
(gdb)

```

Рис. 2.14: Использование команды info breakpoints

```
[ Register Values Unavailable ]

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 96620 In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb) info breakpoints
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x08049000 lab09-2.asm:9
          breakpoint already hit 1 time
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb) █
```

Рис. 2.15: Создание новой точки

```
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x08049000 lab09-2.asm:9
          breakpoint already hit 1 time
2        breakpoint     keep y   0x08049031 lab09-2.asm:20
(gdb)
```

Рис. 2.16: Просмотр информации

```

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd240 0xffffd240

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 96620 In: _start L14 PC: 0x8049016
breakpoint already hit 1 time
2 breakpoint keep y 0x08049031 lab09-2.asm:20
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb)

```

Рис. 2.17: Отслеживание регистров

```

(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb)

```

Рис. 2.18: Просмотр значения переменной

```

(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb)

```

Рис. 2.19: Просмотр значения переменной

```

(gdb) set{char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hello, "
(gdb)

```

Рис. 2.20: Изменение символа

```
(gdb) set{char}0x804a008='h'
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "horld!\n\034"
(gdb)
```

Рис. 2.21: Просмотр значения переменной

```
(gdb) p/t $edx
$1 = 1000
(gdb) p/s $edx
$2 = 8
(gdb) p/x $edx
$3 = 0x8
(gdb)
```

Рис. 2.22: Просмотр значения регистра

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$4 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$5 = 2
(gdb)
```

Рис. 2.23: Изменение регистра командой set

```
(gdb) c
Continuing.
horld!

Breakpoint 2, _start () at lab09-2.asm:20
(gdb) q
```

Рис. 2.24: Прописывание команд c и quit

```
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$
```

Рис. 2.25: Копирование файла


```

odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst
lab09-3.asm
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab
09-3.o
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ gdb --args lab09-3 2 3 '5'

```

Рис. 2.26: Создание и запуск файл в отладчике

```

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 8.
(gdb) run
Starting program: /home/odisupov/work/arch-pc/lab09/lab09-3 2 3 5

Breakpoint 1, _start () at lab09-3.asm:8
8      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd240:      0x00000004
(gdb)

```

Рис. 2.27: Устанавливаем точку

```

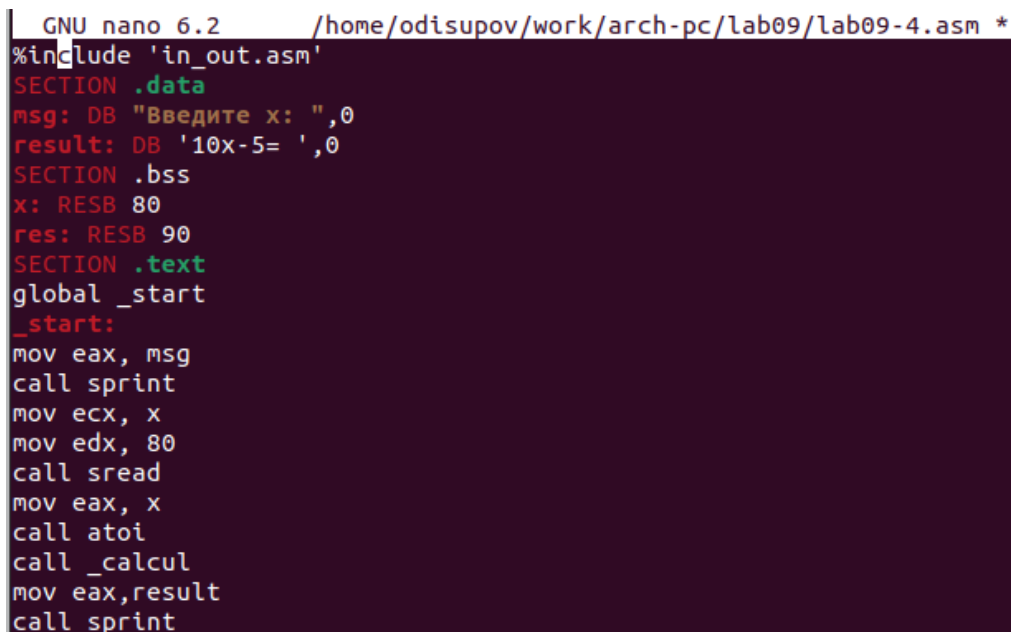
(gdb) x/s *(void**)($esp + 4)
0xffffd3e6:      "/home/odisupov/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)($esp + 8)
0xffffd410:      "2"
(gdb) x/s *(void**)($esp + 12)
0xffffd412:      "3"
(gdb) x/s *(void**)($esp + 16)
0xffffd414:      "5"
(gdb) x/s *(void**)($esp + 20)
0x0:      <error: Cannot access memory at address 0x0>
(gdb)

```

Рис. 2.28: Изучение полученных данных

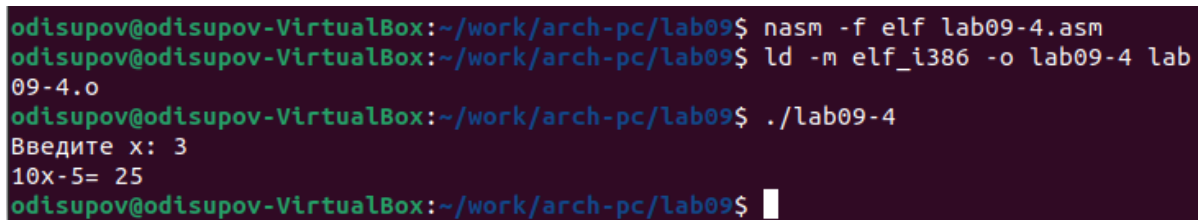
3 Задания для самостоятельной работы

1. Преобразуйте программу из лабораторной работы №8, реализовав вычисление значения функции $\text{fib}(x)$ как подпрограмму



```
GNU nano 6.2 /home/odisupov/work/arch-pc/lab09/lab09-4.asm *
%include 'in_out.asm'
SECTION .data
msg: DB "Введите x: ",0
result: DB '10x-5= ',0
SECTION .bss
x: RESB 80
res: RESB 90
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
```

Рис. 3.1: Копирование и изменение файла



```
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab09-4.asm
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-4 lab09-4.o
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ ./lab09-4
Введите x: 3
10x-5= 25
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$
```

Рис. 3.2: Проверка

2. В листинге 9.3 приведена программа вычисления выражения $(3 + 2) \times 4 + 5$. При запуске данная программа дает неверный результат. Проверьте это. С помощью отладчика GDB, анализируя изменения значений регистров, определите ошибку и исправьте ее

```
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ touch lab09-5.asm
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ mc
```

Рис. 3.3: Создание файла

```
GNU nano 6.2 /home/odisupov/work/arch-pc/lab09/lab09-5.asm *
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
mov ebx,3
mov eax,2
add ebx,eax
mov ecx,4
mul ecx
add ebx,5
mov edi,ebx
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit
```

Рис. 3.4: Изменение файла

```
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ ./lab09-5
Результат: 10
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$
```

Рис. 3.5: Проверка

```

--Register group: general--
eax    0x2      2      ecx    0x4      4      edx    0x0      0
ebx    0x5      5      esp    0xffffd240 0xffffd240  ebp    0x0      0
esi    0x0      0      edi    0x0      0      eip    0x80490f9 0x80490f9 < start+17>
eflags 0x200    0      [ PF IF ]  cs    0x23     35      ss    0x2b     43
ds     0x0      0      es     0x2b     43      fs     0x0      0
gs     0x0      0

B+ 0x80490e0 < start> mov ebx,0x3
0x80490e1 < start+5> mov eax,0x2
0x80490e2 < start+10> add ebx,eax
0x80490e3 < start+12> mov ecx,0x4
> 0x80490f9 < start+17> nul ecx
0x80490f0 < start+19> add ebx,0x5
0x80490f1 < start+22> mov edi,ebx
0x80490f2 < start+24> mov eax,0x004a000
0x80490f3 < start+29> call 0x80490f8 <sprint>
0x80490f4 < start+34> mov eax,edi
0x80490f5 < start+36> call 0x8049000 <iprintLF>
0x80490f6 < start+41> call 0x8049000 <quit>
0x80490f7 < start+44> add BYTE PTR [eax],al
0x80490f8 < start+45> add BYTE PTR [eax],al
0x80490f9 < start+46> add BYTE PTR [eax],al

native process 97438 In: start
(gdb) layout regs
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb)

```

Рис. 3.6: Поиск ошибки регистров в отладчике

```

GNU nano 6.2 /home/odisupov/work/arch-pc/lab09/lab09-5.asm *
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
mov eax,3
mov ebx,2
add eax,ebx
mov ecx,4
mul ecx
add eax,5
mov edi,eax
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit

^G Справка      ^O Записать     ^W Поиск       ^K Вырезать    ^T Выполнить   ^C Позиция
^X Выход        ^R ЧитФайл     ^_ Замена     ^U Вставить    ^J Выводить    ^/ К строке

```

Рис. 3.7: Изменение файла

```
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab
09-5.o
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$ ./lab09-5
Результат: 25
odisupov@odisupov-VirtualBox:~/work/arch-pc/lab09$
```

Рис. 3.8: Проверка

4 Выводы

Я приобрёл навыки написания программ с использованием подпрограмм. Познакомился с методами отладки при помощи GDB и его основными возможностями.