

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра математики и механики

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

дисциплина: Архитектура компьютера

Студент: Кондрацкая Александра Евгеньевна

Группа: НММбд-02-24

МОСКВА

2024 г.

Оглавление

1 Цель работы.....	3
2 Задание.....	4
3 Выполнение лабораторной работы.....	5
4 Выводы.....	19

1 Цель работы

Цель данного задания — освоить практические навыки разработки программ с использованием подпрограмм, а также изучить основы отладки программ с помощью GDB и его ключевые функциональные возможности.

2 Задание

- Освоить реализацию подпрограмм в NASM
- Освоить отладку программ с помощью GDB
- Работа с данными программы GDB
- Обработать аргументы командной строки в GDB

3 Выполнение лабораторной работы

Создадим каталог для выполнения лабораторной работы № 9, перейдём в него и создадим файл lab09-1.asm (рис. 1)

```
aeckondrackaya@dk8n78 ~ $ mkdir ~/work/arch-pc/lab09
aeckondrackaya@dk8n78 ~ $ cd ~/work/arch-pc/lab09
aeckondrackaya@dk8n78 ~/work/arch-pc/lab09 $ touch lab09-1.asm
aeckondrackaya@dk8n78 ~/work/arch-pc/lab09 $ ls
lab09-1.asm
aeckondrackaya@dk8n78 ~/work/arch-pc/lab09 $
```

Рис. 1 Создание каталога

Введём в файл lab09-1.asm текст программы из листинга 9.1 (рис. 2)

```
Открыть
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x:',0
4 result: DB '2x+7=',0
5 SECTION .bss
6 x: RESB 80
7 res: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11 ;-----
12 ; Основная программа
13 ;-----
14 mov eax, msg
15 call sprint
16 mov ecx, x
17 mov edx, 80
18 call sread
19 mov eax, x
20 call atoi
21 call _calcul ; Вызов подпрограммы _calcul
22 mov eax, result
23 call sprint
24 mov eax, [res]
25 call iprintLF
26 call quit
27 ;-----
28 ; Подпрограмма вычисления
29 ; выражения "2x+7"
30 _calcul:
31 mov ebx, 2
32 mul ebx
33 add eax, 7
34 mov [res], eax
35 ret ; выход из подпрограммы
```

Рис. 2 Ввод программы

Создадим исполняемый файл и проверим его работу (рис. 3)

```
aeckondrackaya@dk8n78 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
aeckondrackaya@dk8n78 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
aeckondrackaya@dk8n78 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 2
2x+7=11
aeckondrackaya@dk8n78 ~/work/arch-pc/lab09 $
```

Рис. 3 Исполняемый файл

Внесем изменения в программу (рис. 4+5)

```

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg: DB 'Введите x: ',0
5 prim1: DB 'f(x) = 2x+7',0
6 prim2: DB 'g(x) = 3x-1',0
7 result: DB 'f(g(x))= ',0
8
9 SECTION .bss
10 x: RESB 80
11 res: RESB 80
12
13 SECTION .text
14 GLOBAL _start
15 _start:
16
17 mov eax,prim1
18 call sprintLF
19
20 mov eax,prim2
21 call sprintLF
22
23 mov eax,msg
24 call sprint
25
26 mov ecx,x
27 mov edx,80
28 call sread
29
30 mov eax,x
31 call atoi
32
33 call _calcul
34
35 mov eax,result
36 call sprint
37 mov eax,[res]
38 call iprintLF
39
40 call quit
41
42 _calcul:
43
44 call _subcalcul
45
46 mov ebx,2

```

Рис. 4 изменение программы

```

47 mul ebx
48 add eax,7
49 mov [res],eax
50 ret
51
52 _subcalcul:
53 mov ebx,3
54 mul ebx
55 sub eax,1
56 ret

```

Рис. 5 Изменение в программе

Создадим исполняемый файл и проверим (рис. 6)

```
aekondrackaya@dk8n78 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
aekondrackaya@dk8n78 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
aekondrackaya@dk8n78 ~/work/arch-pc/lab09 $ ./lab09-1
f(x) = 2x+7
g(x) = 3x-1
Введите x: 5
f(g(x))= 35
aekondrackaya@dk8n78 ~/work/arch-pc/lab09 $
```

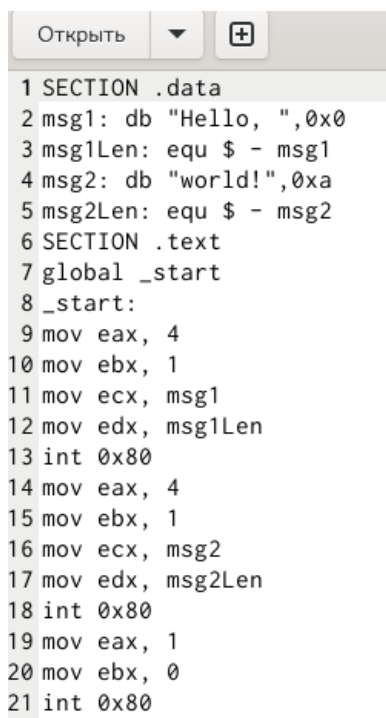
Рис. 6 Исполняемый файл

Создадим файл lab09-2.asm (рис. 7)

```
aekondrackaya@dk8n78 ~/work/arch-pc/lab09 $ touch lab09-2.asm
aekondrackaya@dk8n78 ~/work/arch-pc/lab09 $ ды
bash: ды: команда не найдена
aekondrackaya@dk8n78 ~/work/arch-pc/lab09 $ ls
in_out.asm  lab09-1  lab09-1.asm  lab09-1.o  lab09-2.asm
aekondrackaya@dk8n78 ~/work/arch-pc/lab09 $
```

Рис. 7 Создание файла

Введем в него текст программы из листинга 9.2 (рис. 8)



```
1 SECTION .data
2 msg1: db "Hello, ",0x0
3 msg1Len: equ $ - msg1
4 msg2: db "world!",0xa
5 msg2Len: equ $ - msg2
6 SECTION .text
7 global _start
8 _start:
9 mov eax, 4
10 mov ebx, 1
11 mov ecx, msg1
12 mov edx, msg1Len
13 int 0x80
14 mov eax, 4
15 mov ebx, 1
16 mov ecx, msg2
17 mov edx, msg2Len
18 int 0x80
19 mov eax, 1
20 mov ebx, 0
21 int 0x80
```

Рис. 8 Ввод программы

Создадим исполняемый файл и проверим (рис. 9)

```
aecondrackaya@dk8n78 ~/work/arch-pc/lab09 $ nasm -f elf lab09-2.asm
aecondrackaya@dk8n78 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
aecondrackaya@dk8n78 ~/work/arch-pc/lab09 $ ./lab09-2
Hello, world!
aecondrackaya@dk8n78 ~/work/arch-pc/lab09 $
```

Рис. 9 Исполняемый файл

Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом '-g'. (рис. 10)

```
aecondrackaya@dk8n78 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-2.lst lab09-2.asm
aecondrackaya@dk8n78 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
```

Рис. 10 Добавление отладочной информации

Загрузим исполняемый файл в отладчик gdb и проверим работу программы, запустив ее в оболочке GDB с помощью команды run (рис 11).

```
aecondrackaya@dk8n78 ~/work/arch-pc/lab09 $ gdb lab09-2
GNU gdb (Gentoo 14.2 vanilla) 14.2
Copyright (C) 2023 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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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: /afs/.dk.sci.pfu.edu.ru/home/a/e/aecondrackaya/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 7328) exited normally]
(gdb)
```

Рис. 11 Загрузка в отладчик и команда run

Для более подробного анализа программы установив брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запустив её (рис. 12)

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/e/a:kondrackaya/work/arch-pc/lab09/lab09-2
Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
```

Рис. 12 Брейкпоинт

Посмотрим дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start` (рис. 13)

```
(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.
```

Рис. 13 Просмотр дисассимилированного кода

Переключимся на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel` (рис. 14)

```

(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.

```

Рис. 14 Переключение на другой синтаксис

Главное отличие между этими двумя способами представления команд заключается в использовании символов % и \$ для обозначения регистров в дизассемблированном виде. В Intel-синтаксисе этих символов нет, что упрощает восприятие кода. Включим режим псевдографики для более удобного анализа программы (рис. 15)

```

native process 7652 In: _start
(gdb) layout regs
Undefined command: "layout". Try "help".
(gdb) layout regs
(gdb)

```

0x08049114	add	BYTE PTR [eax], al
0x08049116	add	BYTE PTR [eax], al
0x08049118	add	BYTE PTR [eax], al
0x0804911a	add	BYTE PTR [eax], al
0x0804911c	add	BYTE PTR [eax], al
0x0804911e	add	BYTE PTR [eax], al
0x08049120	add	BYTE PTR [eax], al
0x08049122	add	BYTE PTR [eax], al
0x08049124	add	BYTE PTR [eax], al
0x08049126	add	BYTE PTR [eax], al
0x08049128	add	BYTE PTR [eax], al
0x0804912a	add	BYTE PTR [eax], al
0x0804912c	add	BYTE PTR [eax], al
0x0804912e	add	BYTE PTR [eax], al

Рис. 15 Режим псевдографики

Определим адрес предпоследней инструкции (mov ebx,0x0), установим точку останова и посмотрим информацию о всех установленных точках останова (рис.16)

The screenshot shows the GDB interface. At the top, a message says "[Register Values Unavailable]". Below it, a list of instructions is displayed, each with an address, an opcode, and a description. The instructions are all 'add BYTE PTR [eax], al' at addresses 0x804953a through 0x8049544. Below the instructions, the GDB prompt shows 'native process 7652 In: _start'. Then, the command '(gdb) break *0x8049031' is entered, and the response is 'Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.'. Next, the command '(gdb) i b' is entered, and the response is a table of breakpoints.

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

The GDB prompt '(gdb) ' is shown at the bottom.

Рис. 16 Точки останова

Посмотрим значение переменной msg1 по имени (рис. 17)

The screenshot shows the GDB prompt '(gdb) x/1sb & msg1'. The response is '0x804a000 <msg1>: "Hello, "'.

Рис. 17 Значение переменной по имени

Посмотрим значение переменной msg2 по адресу (рис. 18)

The screenshot shows the GDB prompt '(gdb) x/1sb & msg2'. The response is '0x804a008 <msg2>: "world!\n\034"'. Then, the command '(gdb) x/1sb 0x804a008' is entered, and the response is '0x804a008 <msg2>: "world!\n\034"'. The GDB prompt '(gdb) ' is shown at the bottom.

Рис. 18 Значение переменной по адресу

Изменим первый символ переменной msg1 (рис. 19)

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

Рис. 19 Изменение символа переменной

Заменяем любой символ во второй переменной msg2 (рис. 20)

```
(gdb) set {char}&msg2='u'
(gdb) x/1sb & msg2
0x804a008 <msg2>:      "uor1d!\n\034"
(gdb) █
```

Рис. 20 Изменение символа переменной

С помощью команды set изменим значение регистра ebx (рис. 21)

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$2 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$3 = 2
(gdb) █
```

Рис. 21 Изменение значения регистра

Команда выводит два разных значения, потому что в первый раз указывается значение 2, а во второй раз регистр равен двум, поэтому и значения отличаются.

Скопируем файл lab8-2.asm, созданный при выполнении лабораторной работы №8 (рис. 22)

```
aekondrackaya@dk2n21 ~/work/arch-pc/lab09 $ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
```

Рис. 22 Копирование файла

Создадим исполняемый файл (рис. 23)

```
aecondrackaya@dk2n21 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-3.lst lab09-3.asm
aecondrackaya@dk2n21 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-3 lab09-3.o
```

Рис. 23 Исполняемый файл

Для загрузки в gdb программы с аргументами необходимо использовать ключ --args.

Загрузим исполняемый файл в отладчик, указав аргументы (рис. 24)

```
aecondrackaya@dk2n21 ~/work/arch-pc/lab09 $ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Gentoo 14.2 vanilla) 14.2
Copyright (C) 2023 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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
```

Рис. 24 Загрузка в gdb

Для начала установим точку останова перед первой инструкцией в программе и запустим ее (рис. 25)

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/e/aecondrackaya/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3
Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
```

Рис. 25 Точка останова и запуск

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы) (рис. 26)

```
(gdb) x/x $esp
0xffffc1f0: 0x00000005
```

Рис. 26 Количество аргументов

Их ровно 5.

Посмотрите остальные позиции стека (рис. 27)

```

(gdb) x/s *(void**)(esp + 4)
0xffffc48d:    "/afs/.dk.sci.pfu.edu.ru/home/a/e/aekondrackaya/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffc4d7:    "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffc4e9:    "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffc4fa:    "2"
(gdb) x/s *(void**)(esp + 20)
0xffffc4fc:    "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0:    <error: Cannot access memory at address 0x0>
(gdb) █

```

Рис. 27 Позиции стека

Первый адрес стека содержит указатель на начало данных, а следующие — данные с шагом 4 байта (размер элемента). Для избежания конфликтов, каждое значение записывается в новый стек.

САМОСТОЯТЕЛЬНАЯ РАБОТА

Создадим файл (рис. 28)

```

aekondrackaya@dk8n63 ~ $ cd ~/work/arch-pc/lab09
aekondrackaya@dk8n63 ~/work/arch-pc/lab09 $ touch lab09-4.asm
aekondrackaya@dk8n63 ~/work/arch-pc/lab09 $ ls
in_out.asm  lab09-1.asm  lab09-2      lab09-2.lst  lab09-3      lab09-3.lst  lab09-4.asm
lab09-1     lab09-1.o    lab09-2.asm  lab09-2.o    lab09-3.asm  lab09-3.o
aekondrackaya@dk8n63 ~/work/arch-pc/lab09 $ █

```

Рис. 28 Новый файл

1) Введем измененную программу (рис. 29)


```

1 %include 'in_out.asm'
2
3 SECTION .data
4 prim DB 'f(x)=12x-7',0
5 otv DB 'Результат: ',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9
10 pop ecx
11
12 pop edx
13
14 sub ecx,1
15
16 mov esi,0
17
18 mov eax,prim
19 call sprintLF
20 next:
21 cmp ecx,0
22 jz _end
23
24 pop eax
25 call atoi
26 call fir
27 add esi,eax
28
29 loop next
30
31 _end:
32 mov eax,otv
33 call sprint
34 mov eax,esi
35 call iprintLF
36 call quit
37
38 fir:
39 mov ebx,12
40 mul ebx
41 add eax,7
42 ret

```

Рис. 29 Программа

Создадим исполняемый файл и проверим (рис. 30)

```

aekondrackaya@dk8n63 ~/work/arch-pc/lab09 $ gedit lab09-4.asm
aekondrackaya@dk8n63 ~/work/arch-pc/lab09 $ nasm -f elf lab09-4.asm
aekondrackaya@dk8n63 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-4 lab09-4.o
aekondrackaya@dk8n63 ~/work/arch-pc/lab09 $ ./lab09-4 1 2 3 4
f(x)=12x-7
Результат: 148

```

Рис. 30 Проверка

2) Создадим еще один файл (рис. 31)

```

aekondrackaya@dk8n63 ~/work/arch-pc/lab09 $ touch lab09-5.asm
aekondrackaya@dk8n63 ~/work/arch-pc/lab09 $ ды
bash: ды: команда не найдена
aekondrackaya@dk8n63 ~/work/arch-pc/lab09 $ ls
in_out.asm  lab09-1.asm  lab09-2      lab09-2.lst  lab09-3      lab09-3.lst  lab09-4.asm
lab09-1     lab09-1.o   lab09-2.asm  lab09-2.o   lab09-3.asm  lab09-3.o   lab09-5.asm
aekondrackaya@dk8n63 ~/work/arch-pc/lab09 $

```

Рис. 31 Новый файл

Введем в него программу из листинга (рис. 32)

```

1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 SECTION .text
5 GLOBAL _start
6 _start:
7 ; ---- Вычисление выражения (3+2)*4+5
8 mov ebx,3
9 mov eax,2
10 add ebx,eax
11 mov ecx,4
12 mul ecx
13 add ebx,5
14 mov edi,ebx
15 ; ---- Вывод результата на экран
16 mov eax,div
17 call sprint
18 mov eax,edi
19 call iprintLF
20 call quit

```

Рис. 32 Ввод программы

Создадим исполняемый файл (рис. 33)

```

aekondrackaya@dk8n63 ~/work/arch-pc/lab09 $ nasm -f elf lab09-5.asm
aekondrackaya@dk8n63 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o
aekondrackaya@dk8n63 ~/work/arch-pc/lab09 $ ./lab09-5
Результат: 10

```

Рис. 33 Исполняемый файл

Откроем отладчик и увидим, что регистры стоят не на своих местах (рис. 34)

```
=> 0x080490e8 <+0>: mov ebx, 0x3
0x080490ed <+5>: mov eax, 0x2
0x080490f2 <+10>: add ebx, eax
0x080490f4 <+12>: mov ecx, 0x4
0x080490f9 <+17>: mul ecx
0x080490fb <+19>: add ebx, 0x5
0x080490fe <+22>: mov edi, ebx
0x08049100 <+24>: mov eax, 0x804a000
0x08049105 <+29>: call 0x804900f <sprint>
0x0804910a <+34>: mov eax, edi
0x0804910c <+36>: call 0x8049086 <iprintLF>
0x08049111 <+41>: call 0x80490db <quit>
```

Рис. 34 Отладчик

Исправим это (рис. 35)

```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ', 0
4 SECTION .text
5 GLOBAL _start
6 _start:
7 ; ---- Вычисление выражения (3+2)*4+5
8 mov ebx, 3
9 mov eax, 2
10 add eax, ebx
11 mov ecx, 4
12 mul ecx
13 add eax, 5
14 mov edi, eax
15 ; ---- Вывод результата на экран
16 mov eax, div
17 call sprint
18 mov eax, edi
19 call iprintLF
20 call quit
```

Рис. 35 Исправленная программа

Создадим исполняемый файл и проверим (рис. 36)

```
aecondrackaya@dk8n63 ~/work/arch-pc/lab09 $ nasm -f elf lab09-5.asm
aecondrackaya@dk8n63 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o
aecondrackaya@dk8n63 ~/work/arch-pc/lab09 $ ./lab09-5
Результат: 25
aecondrackaya@dk8n63 ~/work/arch-pc/lab09 $
```

Рис. 36 Проверка

4 Выводы

В результате выполнения задания были приобретены навыки программирования с применением подпрограмм. Изучены базовые методы отладки программ с помощью GDB и освоены его основные функции.