

# **Отчёт по лабораторной работе №7**

**Дисциплина: архитектура компьютера**

Аветисян Алина Эдуардовна

# Содержание

<b>1</b>	<b>Цели работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Выполнение заданий для самостоятельной работы.</b>	<b>14</b>
3.0.1	Задание 1. . . . .	14
3.0.2	Задание 2. . . . .	16
<b>4</b>	<b>Выводы</b>	<b>19</b>

# Список иллюстраций

2.1	Создание каталога и файла. . . . .	6
2.2	Открываю файл lab7-1.asm с помощью текстового редактора nano. . . . .	6
2.3	Исполнение файла. . . . .	7
2.4	Изменение программы. . . . .	7
2.5	Исполнение файла. . . . .	7
2.6	Изменение программы. . . . .	8
2.7	Исполнение файла. . . . .	8
2.8	Создание файла. . . . .	8
2.9	Ввод программы. . . . .	9
2.10	. . . . .	9
2.11	Исполнение файла. . . . .	10
2.12	Создание файла листинга. . . . .	10
2.13	Открываю файл листинга lab7-2.lst с помощью текстового редактора mcedit. . . . .	10
2.14	Файл листинга. . . . .	11
2.15	Файл листинга. . . . .	12
2.16	Файл листинга. . . . .	12
2.17	Трансляция с получением файла листинга. . . . .	13
3.1	Создание файла. . . . .	14
3.2	Редактирование файла. . . . .	15
3.3	. . . . .	15
3.4	Исполнение файла. . . . .	16
3.5	Создание файла. . . . .	16
3.6	Функция $f(x)$ . . . . .	16
3.7	Ввод программы в файл. . . . .	17
3.8	Исполнение файла. . . . .	17
3.9	Исполнение файла. . . . .	18

## **Список таблиц**

# 1 Цели работы

Целями работы являются изучение команд условного и безусловного переходов, приобретение навыков написания программ с использованием переходов, знакомство с назначением и структурой файла листинга.

## 2 Выполнение лабораторной работы

Создаю каталог для программ лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm.

```
aeavetisyan@dk3n37 ~ $ mkdir ~/work/arch-pc/lab07
aeavetisyan@dk3n37 ~ $ cd ~/work/arch-pc/lab07
aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ touch lab7-1.asm
aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $
```

Рис. 2.1: Создание каталога и файла.

Открываю созданный файл lab7-1.asm, вставляю в него программу с использованием инструкции jmp( Листинг 7.1).

```
GNU nano 7.2 /afs/.dk.sci.pfu.edu.ru/home/a/e/aeavetisyan/work/arch-pc/lab07/lab7-1.asm
#include "in_out.asm" ; подключение внешнего файла

SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0

SECTION .text
GLOBAL _start
_start:

jmp _label2

_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 2.2: Открываю файл lab7-1.asm с помощью текстового редактора nano.

Создаю исполняемый файл и запускаю его.

```

aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ █

```

Рис. 2.3: Исполнение файла.

Изменяю текст программы в файле lab7-1.asm в соответствии с Листингом 7.2.

```

GNU nano 7.2 /afs/.dk.sci.pfu.edu.ru/home/a/e/aeavetisyan/work/arch-pc/lab07/lab7-1.asm
#include "in_out.asm" ; подключение внешнего файла

SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0

SECTION .text
GLOBAL _start
_start:

jmp _label2

_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'

_end:
call quit ; вызов подпрограммы завершения

```

Рис. 2.4: Изменение программы.

Создаю исполняемый файл и проверяю его работу.

```

aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ █

```

Рис. 2.5: Исполнение файла.

Изменяю текст программы изменив инструкции jmp.

```

/afs/.dk.sci.pfu.edu.ru/home/a/e/aeavetisyan/work/arch-pc/lab07/lab7-1.asm
#include 'in_out.asm' ; подключение внешнего файла

SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0

SECTION .text
GLOBAL _start
_start:

jmp _label3

_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2

_end:
call quit ; вызов подпрограммы завершения

```

Рис. 2.6: Изменение программы.

Создаю исполняемый файл и проверяю его работу.

```

aeavetisyan@dk5n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
aeavetisyan@dk5n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aeavetisyan@dk5n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
aeavetisyan@dk5n55 ~/work/arch-pc/lab07 $ █

```

Рис. 2.7: Исполнение файла.

Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07.

```

aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ touch lab7-2.asm
aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ █

```

Рис. 2.8: Создание файла.



Ввожу в lab7-2.asm программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C (Листинг 7.3).

```
GNU nano 7.2 /afs/.dk.sci.pfu.edu.ru/home/a/e/aeavetisyan/work/arch-pc/lab07/lab7-2.asm
#include "in_out.asm"
section .data
msg1 db "Введите B: ",0h
msg2 db "Наибольшее число: ",0h
; dd '20'
; dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax,msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintf ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рис. 2.9: Ввод программы.

```
GNU nano 7.2 /afs/.dk.sci.pfu.edu.ru/home/a/e/aeavetisyan/work/arch-pc/lab07/lab7-2.asm
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax,msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintf ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рис. 2.10:

Создаю исполняемый файл и проверяю его работу для разных значений B: 35 и 70.

```

aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ touch lab7-2.asm
aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 35
Наибольшее число: 50
aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 70
Наибольшее число: 70
aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ █

```

Рис. 2.11: Исполнение файла.

Создаю файл листинга для программы из файла lab7-2.asm.

```

aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm

```

Рис. 2.12: Создание файла листинга.

Открываю файл листинга lab7-2.lst с помощью mcedit.

```

aeavetisyan@dk3n37 ~/work/arch-pc/lab07 $ mcedit lab7-2.lst

```

Рис. 2.13: Открываю файл листинга lab7-2.lst с помощью текстового редактора mcedit.

Рассмотрим 5, 9 и 10 строки.

```

/home/aeavetisyan/work/arch-pc/lab07/lab7-2.lst [----] 0 L: 1+ 0 1/225] *(0 /14458b) 0032 0x020
1 %include 'in_out.asm'
2 <1> ;----- slen -----
3 <1> ; Функция вычисления длины сообщения
4 <1> slen:
5 00000000 53 <1> push ebx
6 00000001 89C3 <1> mov ebx, eax
7 <1>
8 <1> nextchar:
9 00000003 003800 <1> cnp byte [eax], 0
10 00000006 7403 <1> jz finished
11 00000008 40 <1> inc eax
12 00000009 EBF8 <1> jmp nextchar
13 <1>
14 <1> finished:
15 0000000B 29D8 <1> sub eax, ebx
16 0000000D 5B <1> pop ebx
17 0000000E C3 <1> ret
18 <1>
19 <1>
20 <1> ;----- sprintf -----
21 <1> ; Функция печати сообщения
22 <1> ; входные данные: mov eax,<message>
23 <1> sprintf:
24 0000000F 52 <1> push edx
25 00000010 51 <1> push ecx
26 00000011 53 <1> push ebx
27 00000012 50 <1> push eax
28 00000013 E8E8FFFFFF <1> call slen
29 <1>
30 00000018 89C2 <1> mov edx, eax
31 0000001A 5B <1> pop eax
32 <1>
33 0000001B 89C1 <1> mov ecx, eax
34 0000001D B801000000 <1> mov ebx, 1
35 00000022 B804000000 <1> mov eax, 4
36 00000027 CD00 <1> int 80h
37 <1>
38 00000029 5B <1> pop ebx
39 0000002A 59 <1> pop ecx
40 0000002B 5A <1> pop edx
41 0000002C C3 <1> ret
42 <1>
43 <1>
44 <1> ;----- sprintfLF -----
45 <1> ; Функция печати сообщения с переводом строки
46 <1> ; входные данные: mov eax,<message>
47 <1> sprintfLF:
48 0000002D E8D0FFFFFF <1> call sprintf
49 <1>
50 00000032 50 <1> push eax
51 00000033 B80A000000 <1> mov eax, 0Ah
52 00000038 50 <1> push eax
53 00000039 89E0 <1> mov eax, esp

```

Рис. 2.14: Файл листинга.

5 строка:

- Первые цифры [5] - это номер строки файла листинга.
- Следующие цифры [00000000] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.
- Следующие числа [53] - это машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, поэтому и появляются буквы латинского алфавита.
- Следующее [push ebx] - исходный текст программы, которая просто состоит из строки исходной программы вместе с комментариями.

9 строка:

- Первые цифры [9] - это номер строки файла листинга.

- Следующие цифры [00000006] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.

- Следующие числа [7403] - это машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, поэтому и появляются буквы латинского алфавита.

- Следующее [jz finished] - исходный текст программы, которая просто состоит из строки исходной программы вместе с комментариями.

10 строка:

- Первые цифры [10] - это номер строки файла листинга.

- Следующие цифры [00000008] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.

- Следующие числа [40] - это машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, поэтому и появляются буквы латинского алфавита.

- Следующее [inc eax] - исходный текст программы, которая просто состоит из строки исходной программы вместе с комментариями.

Открываю файл lab7-2.lst с помощью редактора и удаляю один операнд в инструкции `cmp`.

```
38 mov ecx,[max]
39 cmp ecx, ; Сравниваем 'max(A,C)' и 'B'
40 jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 mov ecx,[B] ; иначе 'ecx = B'
42 mov [max],ecx
```

Рис. 2.15: Файл листинга.

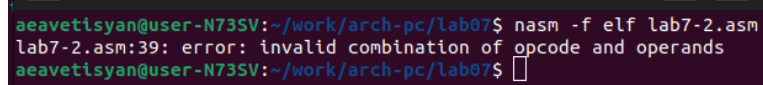
Открываю файл листинга с помощью редактора `mcedit` и замечаю, что в файле листинга появляется ошибка.

```
39          *****      cmp ecx, ; Сравниваем 'max(A,C)' и 'B'
                                     error: invalid combination of opcode and operands
40 00000145 7F0C          jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 00000147 8B00[0A000000]  mov ecx,[B] ; иначе 'ecx = B'
42 0000014D 8900[00000000]  mov [max],ecx
43                                     ; ----- Вывод результата
44
```

Рис. 2.16: Файл листинга.

Отсюда можно сделать вывод, что, если в коде появляется ошибка, то ее описание появится в файле листинга.

Выполняю трансляцию с получением файла листинга.



```
aeavetisyan@user-N73SV:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
lab7-2.asm:39: error: invalid combination of opcode and operands
aeavetisyan@user-N73SV:~/work/arch-pc/lab07$
```

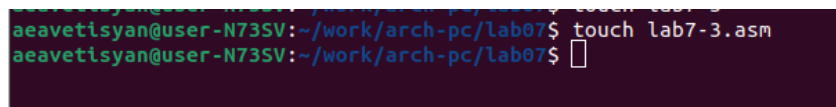
Рис. 2.17: Трансляция с получением файла листинга.

Создаётся файл листинга, в котором есть ошибка.

## 3 Выполнение заданий для самостоятельной работы.

### 3.0.1 Задание 1.

Создаю файл lab7-3.asm с помощью утилиты touch.



```
aeavetisyan@user-N73SV:~/work/arch-pc/lab07$ touch lab7-3.asm
aeavetisyan@user-N73SV:~/work/arch-pc/lab07$
```

Рис. 3.1: Создание файла.

Ввожу в созданный файл текст программы для нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных беру, учитывая свой вариант из прошлой лабораторной работы, 15 вариант.

```

GNU nano 6.2 /home/aeavetisyan/work/arch-pc/lab07/lab7-3.asm *
#include 'in_out.asm'
section .data
msg1 db 'a = ',0h
msg2 db 'b = ',0h
msg3 db 'c = ',0h
msg4 db "Наименьшее число: ",0h
a dd '32'
b dd '16'
c dd '54'

section .bss
max resb 10

section .text
global _start
_start:
; ----- Вывод всех чисел:
mov eax,msg1
call sprint
mov eax,a
call atoi
call fprintf

mov eax,msg2
call sprint
mov eax,b
call atoi
call fprintf

mov eax,msg3
call sprint
mov eax,c
call atoi
call fprintf

;-----сравнение чисел
mov eax,a
call atoi ;перевод символа в число
mov [a],eax ; запись преобразованного числа в b
;----- запись b в переменную max
mov eax,b
call atoi ;перевод символа в число
mov [b],eax ; запись преобразованного числа в b

mov eax,c
call atoi ;перевод символа в число
mov [c],eax ; запись преобразованного числа в b
mov ecx,[a] ;

```

Рис. 3.2: Редактирование файла.

```

GNU nano 6.2 /home/aeavetisyan/work/arch-pc/lab07/lab7-3.asm *
mov eax,msg2
call sprint
mov eax,b
call atoi
call fprintf

mov eax,msg3
call sprint
mov eax,c
call atoi
call fprintf

;-----сравнение чисел
mov eax,a
call atoi ;перевод символа в число
mov [a],eax ; запись преобразованного числа в b
;----- запись b в переменную max
mov eax,b
call atoi ;перевод символа в число
mov [b],eax ; запись преобразованного числа в b

mov eax,c
call atoi ;перевод символа в число
mov [c],eax ; запись преобразованного числа в b
mov ecx,[a] ;
mov [max],ecx ;
;-----сравнение чисел a с
cmp ecx,[c]; if asc
jl check_b ; то переход на метку
mov ecx,[c] ; else ecx=c
mov [max],ecx ; max=c
;-----метка check_b
check_b:
;-----
mov ecx,[max] ;
cmp ecx,[b] ; ecx=b
jl check_c ;
mov ecx,[b] ;
mov [max],ecx ;
;-----
check_c:
mov eax,msg4 ;
call sprint ;
mov eax,[max];
call fprintf ;
call quit

```

Рис. 3.3:

Создаю исполняемый файл и запускаю его.

```

aeavetisyan@user-N735V:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
aeavetisyan@user-N735V:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
aeavetisyan@user-N735V:~/work/arch-pc/lab07$ ./lab7-3
a = 32
b = 6
c = 54
Наименьшее число: 6
aeavetisyan@user-N735V:~/work/arch-pc/lab07$ 

```

Рис. 3.4: Исполнение файла.

### 3.0.2 Задание 2.

Создаю новый файл lab7-4 с помощью утилиты touch.

```

aeavetisyan@user-N735V:~/work/arch-pc/lab07$ touch lab7-4.asm
aeavetisyan@user-N735V:~/work/arch-pc/lab07$ 

```

Рис. 3.5: Создание файла.

Функцию беру из таблицы в соответствии со своим вариантом.

$$15 \quad \begin{cases} a + 10, & x < a \\ x + 10, & x \geq a \end{cases} \quad (2;3) \quad (4;2)$$

Рис. 3.6: Функция f(x).

Ввожу в lab7-4.asm программу, в которую ввожу 2 значения x и a, и которая выводит значения функции.



```

GNU nano 6.2 /home/aeavetisyan/work/arch-pc/lab07/lab7-4.asm
#include 'in_out.asm'
section .data
msg1 db 'Введите x: ',0h
msg2 db 'Введите a: ',0h
msg3 db 'f(x) = ',0h

section .bss
x resb 10
a resb 10

section .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,x
mov edx,10
call sread
mov eax,x
;-----
call atoi
mov [x],eax
;-----

mov eax,msg2
call sprint
mov ecx,a
mov edx,10
call sread
mov eax,a ;
call atoi
mov [a],eax ;
;-----
mov ecx,[x]
cmp ecx,[a] ;x<a
jl _check ;
mov ecx,[x]
add ecx,10
jmp _end
_check:
mov ecx,[a];
add ecx,10
_end:
mov eax,msg3 ;
call sprint ;
mov eax,ecx ;
call iprintLF;
call quit ;

```

Рис. 3.7: Ввод программы в файл.

Создаю исполняемый файл и проверяю его выполнение при  $x=2$ ,  $a=3$ . Программа отработала верно.

```

aeavetisyan@user-N73SV:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
aeavetisyan@user-N73SV:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
aeavetisyan@user-N73SV:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 2
Введите a: 3
f(x) = 13

```

Рис. 3.8: Исполнение файла.

Повторный раз запускаю программу и проверяю его выполнение при  $x=4$  и

a=2. Программа отработала верно.

```
aeavetisyan@user-N73SV:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
aeavetisyan@user-N73SV:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
aeavetisyan@user-N73SV:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 4
Введите a: 2
f(x) = 14
aeavetisyan@user-N73SV:~/work/arch-pc/lab07$
```

Рис. 3.9: Исполнение файла.

## 4 Выводы

При выполнении данной лабораторной работы я освоила инструкции условного и безусловного вывода и ознакомилась с структурой файла листинга.