

Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютера

Арфонос Дмитрий

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Сомнительная работа	13
4	Выводы	21

Список иллюстраций

2.1	Создание директории	5
2.2	Создание копии файла для дальнейшей работы	5
2.3	Редактирование файла	6
2.4	Запуск исполняемого файла	6
2.5	Редактирование файла	7
2.6	Запуск исполняемого файла	7
2.7	Редактирование программы	8
2.8	Создание исполняемого файла	8
2.9	Создание файла	8
2.10	Вставляю текст в файл	9
2.11	Запуск исполняемого файла	9
2.12	Редактирование файла	10
2.13	Файл листинга	10
2.14	Файл листинга	12
2.15	Файл листинга	12
3.1	Создание файла	13
3.2	Редактирование файла	14
3.3	Запуск исполняемого файла	14
3.4	создание файла	17
3.5	функция $f(x)$	17
3.6	ввод программы в файл	18
3.7	Создание исполняемого файла	18
3.8	запуск исполняемого файла	19

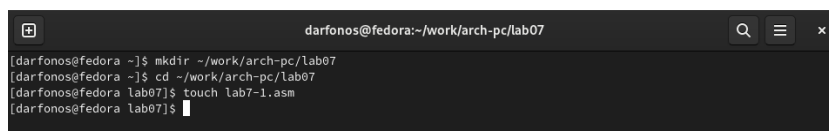
1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга

2 Выполнение лабораторной работы

1

С помощью утилиты `mkdir` создаю директорию `lab07`, перехожу в нее и создаю файл для работы. (рис. [2.1])



```
darfonos@fedora:~/work/arch-pc/lab07
[darfonos@fedora ~]$ mkdir ~/work/arch-pc/lab07
[darfonos@fedora ~]$ cd ~/work/arch-pc/lab07
[darfonos@fedora lab07]$ touch lab7-1.asm
[darfonos@fedora lab07]$
```

Рис. 2.1: Создание директории

2

Копирую в текущий каталог файл `in_out.asm` из загрузок, т.к. он будет использоваться в других программах (рис. [2.2]).

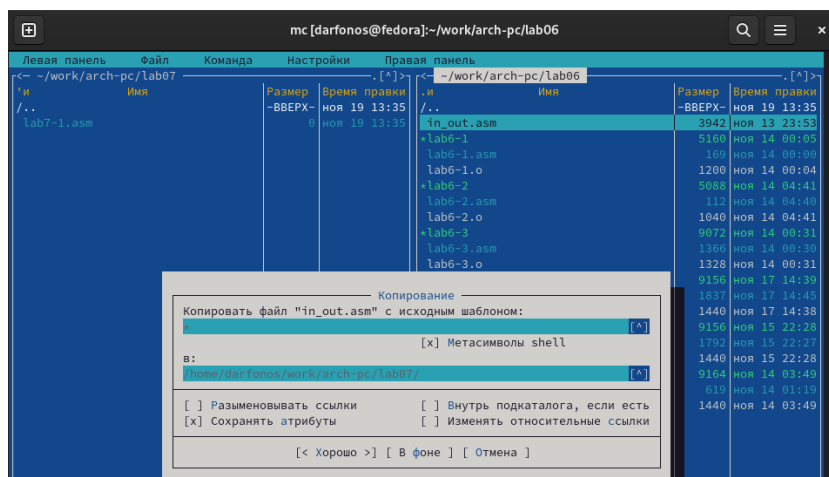


Рис. 2.2: Создание копии файла для дальнейшей работы

3

Открываю созданный файл lab7-1.asm, вставляю в него программу реализации безусловных переходов(рис. [2.3]).



```
darfonos@fedora:~/work/arch-pc/lab07
GNU nano 7.2 /home/darfonos/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. 2.3: Редактирование файла

4

Создаю исполняемый файл программы и запускаю его (рис. [2.4]). Инструкции jmp _label2 меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки _label2.



```
[darfonos@fedora lab07]$ nasm -f elf lab7-1.asm
[darfonos@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[darfonos@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[darfonos@fedora lab07]$
```

Рис. 2.4: Запуск исполняемого файла

5

Изменяю текст программы так, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу (рис. [2.5]).

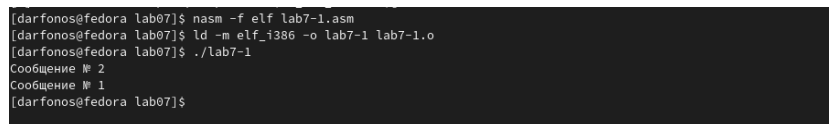


```
GNU nano 7.2 /home/darfonos/work/arch-pc/lab07/lab7-1.asm
#include "in_out.asm" ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
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. 2.5: Редактирование файла

6

Создаю новый исполняемый файл программы и запускаю его (рис. [2.6]). Убеждаюсь в том, программа работает верно.

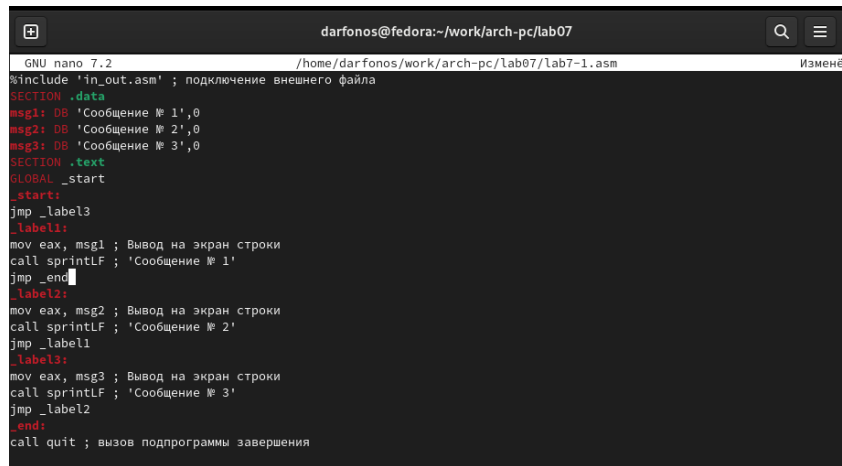


```
[darfonos@fedora lab07]$ nasm -f elf lab7-1.asm
[darfonos@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[darfonos@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 1
Сообщение № 3
[darfonos@fedora lab07]$
```

Рис. 2.6: Запуск исполняемого файла

7

Изменяю текст программы, так чтобы вывод происходил в обратном порядке (рис. [2.7]).



```
GNU nano 7.2 /home/darfonos/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. 2.7: Редактирование программы

8

Создаю исполняемый файл и проверяю работу программы (рис. [2.8]). Программа отработало верно.



```
[darfonos@fedora lab07]$ nasm -f elf lab7-1.asm
[darfonos@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[darfonos@fedora lab07]$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[darfonos@fedora lab07]$
```

Рис. 2.8: Создание исполняемого файла

9

Создаю новый файл lab7-2.asm для программы с условным оператором. (рис. [2.9]).



```
[darfonos@fedora lab07]$ touch lab7-2.asm
[darfonos@fedora lab07]$ mc
```

Рис. 2.9: Создание файла

10

Вставляю программу, которая определяет и выводит на экран наибольшее число (рис.[2.10]).


```
darfonos@fedora:~/work/arch-pc/lab07
GNU nano 7.2 /home/darfonos/work/arch-pc/lab07/lab7-2.asm
#include "in_out.asm"
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C 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
```

Рис. 2.10: Вставляю текст в файл

11

Создаю и запускаю новый исполняемый файл, проверяю работу программы для разных B=40 и B=60, при A=20 и C=50 (рис. [2.11]).

```
[darfonos@fedora lab07]$ nasm -f elf lab7-2.asm
[darfonos@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[darfonos@fedora lab07]$ ./lab7-2
Введите B: 40
Наибольшее число: 50
[darfonos@fedora lab07]$ ./lab7-2
Введите B: 60
Наибольшее число: 60
[darfonos@fedora lab07]$
```

Рис. 2.11: Запуск исполняемого файла

12

Создаю файл листинга для программы в файле lab7-2.asm (рис. [2.12]).

```
darfonos@fedora:~/work/arch-pc/lab07
[darfonos@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
[darfonos@fedora lab07]$ mcedit lab7-2.lst
```

Рис. 2.12: Редактирование файла

13

Открываю файл листинга с помощью редактора mcedit. Рассмотрим 9-11 строки: (рис. [2.13]).

```
lab7-2.lst  [----]  0 L: [ 1+ 0  1/225] *(0  /14458b) 0032 0x020  [*][X]
1  %include 'in_out.asm'
2  <1> ;----- slen -----
3  <1> ; Функция вычисления длины сообщения
4  <1> slen:.....
5  <1>   push    ebx.....
6  <1>   mov     ebx, eax.....
7  <1> .....
8  <1> nextchar:.....
9  <1>   cmp     byte [eax], 0...
10 <1>   jz      finished.....
11 <1>   inc     eax.....
12 <1>   jmp     nextchar.....
13 <1> .....
14 <1> finished:.....
15 <1>   sub     eax, ebx
16 <1>   pop     ebx.....
17 <1>   ret.....
18 <1> .....
19 <1> ;----- sprint -----
20 <1> ; Функция печати сообщения
21 <1> ; входные данные: mov eax,<message>
22 <1> sprint:.....
23 <1>   push    edx
24 <1>   push    ecx
25 <1>   push    ebx
26 <1>   push    eax
27 <1>   call   slen
28 <1> .....
29 <1>   mov     edx, eax
30 <1>   pop     eax
31 <1> .....
32 <1>   mov     ecx, eax
33 <1>   mov     ebx, 1
34 <1>   mov     eax, 4
35 <1>   int     80h
36 <1> .....
37 <1>   pop     ebx
1 Помощь  2 Сохранить  3 Блок  4 Замена  5 Копия  6 Перейти  7 Поиск  8 Удалить  9 МенюМС  10 Выход
```

Рис. 2.13: Файл листинга

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 строк 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 строк исходной программы вместе с комментариями

11 строка:

- Первые цифры [11] - это номер строки файла листинга.
- Следующие цифры [00000009] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.
- следующие числа [EBF8] - это машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, поэтому и появляются буквы латинского алфавита.
- следующее [jmp nextchar] - исходный текст программы, которая просто состоит из строк исходной программы вместе с комментариями

14

Открываю файл lab7-2.asm с помощью редактора и Удаляю один операнд в инструкции str. (рис. [2.14]).

```

cmp ecx ; 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:

```

Рис. 2.14: Файл листинга

15

Открываю файл листинга с помощью редактора mscedit и замечаю, что в файле листинга появляется ошибка. (рис. [2.15]).

```

28      cmp ecx ; Сравниваем 'A' и 'C'
28      error: invalid combination of opcode and operands
29      jg check_B ; если 'A>C', то переход на метку 'check_B',
30      mov ecx,[C] ; иначе 'ecx = C'
31      mov [max],ecx ; 'max = C'
32      ; ----- Преобразование 'max(A,C)' из символа в число
33      check_B:
34      mov eax,max
35      call atoi ; Вызов подпрограммы перевода символа в число
36      mov [max],eax ; запись преобразованного числа в 'max'
37      ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)

```

Рис. 2.15: Файл листинга

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

3 Сомтоятельная работа

1

Создаю файл lab7-3.asm с помощью утилиты touch (рис. [3.1]).

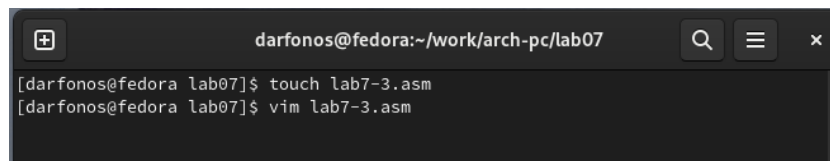
A screenshot of a terminal window with a dark background. The title bar at the top shows the user 'darfonos' on a 'fedora' machine, in the directory '~/work/arch-pc/lab07'. The terminal contains two lines of text: the first line shows the command 'touch lab7-3.asm' being executed, and the second line shows the command 'vim lab7-3.asm' being entered. The prompt character is '\$'.

Рис. 3.1: Создание файла

2

Ввожу в созданный файл текст программы для вычисления наименьшего из 3 чисел. Числа беру, учитывая свой вариант из прошлой лабораторной работы. 2 вариант (рис. [3.2]).

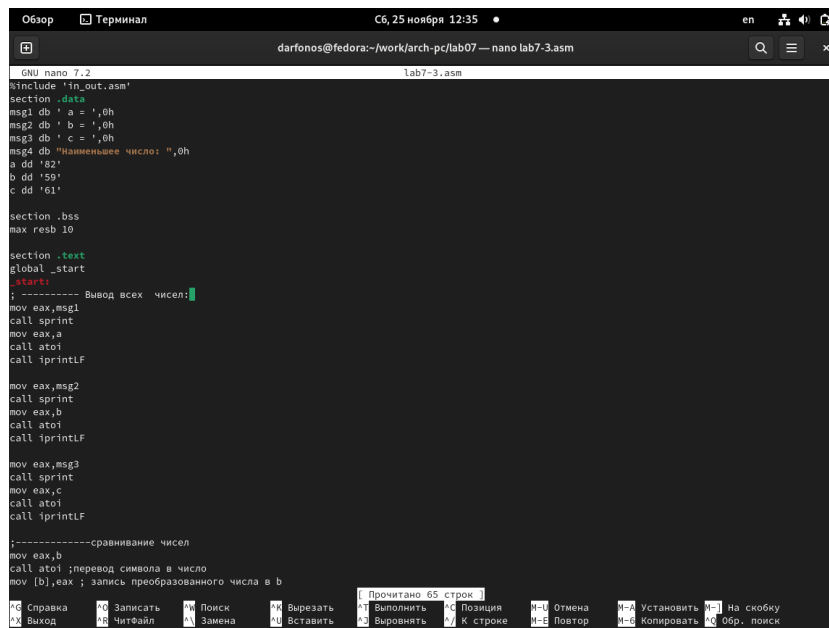


Рис. 3.2: Редактирование файла

3

Создаю исполняемый файл и запускаю его (рис. [3.3]).

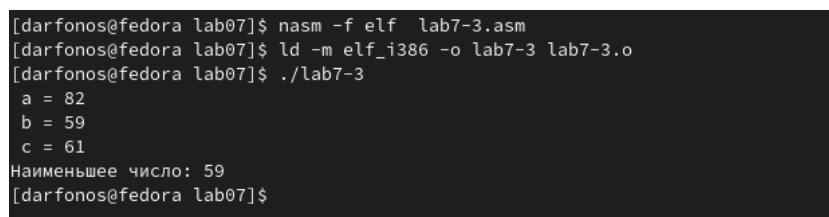


Рис. 3.3: Запуск исполняемого файла

Текст программы

```

#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 '82'
b dd '59'
c dd '61'
```

```
section .bss
max resb 10
```

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

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

mov eax,msg3
call sprint
mov eax,c
call atoi
call iprintLF
```

```

;-----сравнивание чисел
mov eax,b
call atoi ;перевод символа в число
mov [b],eax ; запись преобразованного числа в b
;----- запись b в переменную max
mov ecx,[a] ;
mov [max],ecx ;
;-----сравнивание чисел a c
cmp ecx,[c]; if a>c
jl check_b ; то перход на метку
mov ecx,[c] ;
mov [max],ecx ;
;-----метка check_b
check_b:
mov eax,max ;
call atoi
mov [max],eax ;
;-----
mov ecx,[max] ;
cmp ecx,[b] ;
jl check_c ;
mov ecx,[b] ;
mov [max],ecx ;
;-----
check_c:
mov eax,msg4 ;
call sprint ;
mov eax,[max];

```



```
call iprintLF ;  
call quit
```

4

Создаю новый файл lab7-4 для написания программы второго задания. (рис. [3.4]).

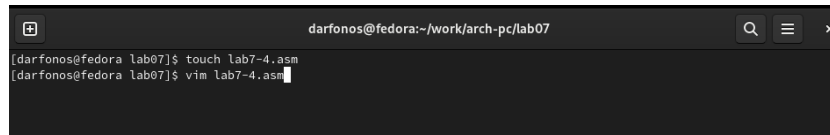


Рис. 3.4: создание файла

5

Ввожу в него программу, (рис. [3.6]). в которую ввожу 2 значения x и a , и которая выводит значения функции. Функцию беру из таблицы в соответствии со своим вариантом (Вариант рис. [3.5]).

$$2 \quad \begin{cases} a-1, & x < a \\ x-1, & x \geq a \end{cases} \quad (5;7) \quad (6;4)$$

Рис. 3.5: функция $f(x)$

```
GNU nano 7.2 /home/darfonos/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,[a]
cmp ecx,[x] ;x<a
jg check_a ;

[ Прочитано 44 строки ]
^G Справка      ^O Записать     ^W Поиск        ^K Вырезать     ^T Выполнить    ^C Позиция
^X Выход        ^R ЧитФайл     ^\ Замена       ^U Вставить     ^J Выводить     ^_ К строке
```

Рис. 3.6: ввод программы в файл

6

Создаю исполняемый файл и проверяю её выполнение при $x=5$, $a=7$ (рис. [3.7]).

Программа отработала верно!

```
[darfonos@fedora lab07]$ nasm -f elf lab7-4.asm
[darfonos@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
[darfonos@fedora lab07]$ ./lab7-4
Введите x: 5
Введите a: 7
f(x) = 6
[darfonos@fedora lab07]$
```

Рис. 3.7: Создание исполняемого файла

7

Повторный раз запускаю программу и проверяю ее выполнение при $x=6$ и $a=4$ (рис. [3.8]). Программа отработала верно!

```
[darfonos@fedora lab07]$ ./lab7-4
Введите x: 6
Введите a: 4
f(x) = 5
[darfonos@fedora lab07]$
```

Рис. 3.8: запуск исполняемого файла

Текст программы

```
%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,[a]
cmp ecx,[x] ;x<a
jg check_a ;
mov ecx,[x]
check_a:
add ecx,-1;
mov eax,msg3 ;
call sprint ;
mov eax,ecx ;
call iprintLF;
call quit ;

```

4 Выводы

При выполнении данной лабораторной работы я освоил инструкции условного и безусловного вывода и ознакомился с структурой файла листинга.ы