

Отчёт по лабораторной работе №7

**Команды безусловного и условного перехода в NASM.
Программирование ветвлений**

Новикова Анастасия Андреевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Реализация переходов в NASM	7
3.2	Изучение структуры файлы листинга	10
3.3	Задания для самостоятельной работы	12
4	Выводы	19

Список иллюстраций

3.1	Создание файлов для лабораторной работы	7
3.2	Ввод текста программы из листинга 7.1	7
3.3	Запуск программного кода	8
3.4	Изменение текста программы	8
3.5	Создание исполняемого файла	8
3.6	Изменение текста программы	9
3.7	Вывод программы	9
3.8	Создание файла	9
3.9	Ввод текста программы из листинга 7.3	10
3.10	Проверка работы файла	10
3.11	Создание файла листинга	10
3.12	Изучение файла листинга	11
3.13	Выбранные строки файла	11
3.14	Удаление выделенного операнда из кода	12
3.15	Получение файла листинга	12
3.16	Ошибка в файле листинга	12
3.17	Написание программы	13
3.18	Запуск файла и проверка его работы	13
3.19	Написание программы	15
3.20	Запуск файла и проверка его работы	16

Список таблиц

1 Цель работы

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

2 Задание

1. Реализация переходов в NASM.
2. Изучение структуры файлы листинга.
3. Задания для самостоятельной работы.

3 Выполнение лабораторной работы

3.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm. (рис. 3.1).

```
aanovikova123@fedora:~$ mkdir ~/work/arch-pc/lab07
aanovikova123@fedora:~$ cd ~/work/arch-pc/lab07
aanovikova123@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
aanovikova123@fedora:~/work/arch-pc/lab07$
```

Рис. 3.1: Создание файлов для лабораторной работы

Ввожу в файл lab7-1.asm текст программы из листинга 7.1. ((рис. 3.2).



```
Открыть  lab7-1.asm
~/work/arch-pc/lab07
1 %include "in_out.asm" ; подключение внешнего файла
2 SECTION .data
3 msg1: DB "Сообщение № 1",0
4 msg2: DB "Сообщение № 2",0
5 msg3: DB "Сообщение № 3",0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; "Сообщение № 1"
13 _label2:
14 mov eax, msg2 ; Вывод на экран строки
15 call sprintf ; "Сообщение № 2"
16 _label3:
17 mov eax, msg3 ; Вывод на экран строки
18 call sprintf ; "Сообщение № 3"
19 _end:
20 call quit ; вызов подпрограммы завершения
```

Рис. 3.2: Ввод текста программы из листинга 7.1

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

```

aanovikova123@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aanovikova123@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
aanovikova123@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
aanovikova123@fedora:~/work/arch-pc/lab07$

```

Рис. 3.3: Запуск программного кода

Таким образом, использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения.

Изменяю программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого изменяю текст программы в соответствии с листингом 7.2. (рис. 3.4)

```

Открыть [icon] *lab7-1.asm
~/work/arch-pc/lab07
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintf ; 'Сообщение № 3'
21 _end:
22 call quit ; вызов подпрограммы завершения

```

Рис. 3.4: Изменение текста программы

Создаю исполняемый файл и проверяю его работу. (рис. 3.5)

```

aanovikova123@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aanovikova123@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1-1
aanovikova123@fedora:~/work/arch-pc/lab07$ ./lab7-1-1
Сообщение № 2
Сообщение № 1
aanovikova123@fedora:~/work/arch-pc/lab07$

```

Рис. 3.5: Создание исполняемого файла

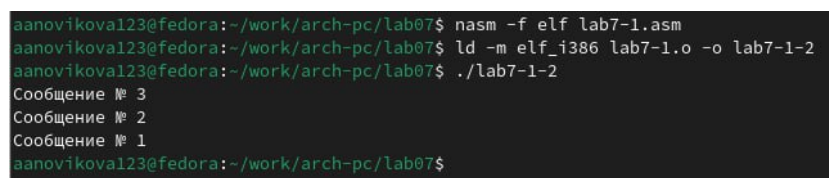
Затем изменяю текст программы, добавив в начале программы `jmp _label3`, `jmp _label2` в конце метки `jmp _label3`, `jmp _label1` добавляю в конце метки `jmp _label2`, и добавляю `jmp _end` в конце метки `jmp _label1`, (рис. 3.6)



```
1 %include "in_out.asm" ; подключение внешнего файла
2 SECTION .data
3 msg1: DB "Сообщение № 1",0
4 msg2: DB "Сообщение № 2",0
5 msg3: DB "Сообщение № 3",0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label3
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call printf ; "Сообщение № 1"
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call printf ; "Сообщение № 2"
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call printf ; "Сообщение № 3"
21 jmp _label2
22 _end:
23 call quit ; вызов подпрограммы завершения
```

Рис. 3.6: Изменение текста программы

чтобы вывод программы был следующим: (рис. 3.7)

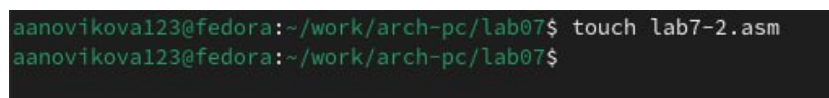


```
aanovikova123@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aanovikova123@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1-2
aanovikova123@fedora:~/work/arch-pc/lab07$ ./lab7-1-2
Сообщение № 3
Сообщение № 2
Сообщение № 1
aanovikova123@fedora:~/work/arch-pc/lab07$
```

Рис. 3.7: Вывод программы

Рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А,В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры.

Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. (рис. 3.8)



```
aanovikova123@fedora:~/work/arch-pc/lab07$ touch lab7-2.asm
aanovikova123@fedora:~/work/arch-pc/lab07$
```

Рис. 3.8: Создание файла

Текст программы из листинга 7.3 ввожу в lab7-2.asm. ((рис. 3.9)

```

Открыть  lab7-2.asm
~work/arch-pc/lab07

1 include "in_out.asm"
2 section .data
3 msg1 db "Введите B: ",0h
4 msg2 db "Наибольшее число: ",0h
5 A dd "20"
6 C dd "50"
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения "Введите B: "
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод "B"
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование "B" из символа в число
21 mov eax,B
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в "B"
24 ; ----- Записываем "A" в переменную "max"
25 mov ecx,[A] ; "ecx = A"
26 mov [max],ecx ; "max = A"
27 ; ----- Сравниваем "A" и "C" (как символы)
28 mov ecx,[C] ; Сравниваем "A" и "C"
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)
38 mov ecx,[max]
39 mov ecx,[B] ; 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
43 ; ----- Вывод результата
44 fin:
45 mov eax,msg2
46 call sprint ; Вывод сообщения "Наибольшее число: "
47 mov eax,[max]
48 call sprintf ; Вывод "max(A,B,C)"
49 call quit ; Выход

```

Рис. 3.9: Ввод текста программы из листинга 7.3

Создаю исполняемый файл и проверю его работу. (рис. 3.10)

```

aanovikova123@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aanovikova123@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
aanovikova123@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 15
Наибольшее число: 50
aanovikova123@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 68
Наибольшее число: 68

```

Рис. 3.10: Проверка работы файла

Файл работает корректно.

3.2 Изучение структуры файла листинга

Создаю файл листинга для программы из файла lab7-2.asm. (рис. 3.11)

```

Наибольшее число: 68
aanovikova123@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm

```

Рис. 3.11: Создание файла листинга

Открываю файл листинга lab7-2.lst с помощью текстового редактора и внимательно изучаю его формат и содержимое. (рис. 3.12)

```

Открыть  lab7-2.lst
~\test\lab7-2\lab07

1 1 %include 'in_out.asm'
2 1 <1> ;----- slen -----
3 2 <1> ; Функция вычисления длины сообщения
4 3 <1> slen:
5 4 00000000 53 <1> push ebx
6 5 00000001 89C3 <1> mov ebx, eax
7 6 <1>
8 7 <1> nextchar:
9 8 00000003 803800 <1> cmp byte [eax], 0
10 9 00000006 7403 <1> jz finished
11 10 00000008 40 <1> inc eax
12 11 00000009 EBF8 <1> jmp nextchar
13 12 <1>
14 13 <1> finished:
15 14 00000008 29D8 <1> sub eax, ebx
16 15 00000000 5B <1> pop ebx
17 16 0000000E C3 <1> ret
18 17 <1>
19 18 <1>
20 19 <1> ;----- sprintf -----
21 20 <1> ; Функция печати сообщения
22 21 <1> ; входные данные: mov eax, <message>
23 22 <1> sprintf:
24 23 0000000F 52 <1> push edx
25 24 00000010 51 <1> push ecx
26 25 00000011 53 <1> push ebx
27 26 00000012 50 <1> push eax
28 27 00000013 EBEBFFFFFF <1> call slen
29 28 <1>
30 29 00000018 89C2 <1> mov edx, eax
31 30 0000001A 58 <1> pop eax
32 31 <1>
33 32 0000001B 89C1 <1> mov ecx, eax
34 33 0000001D 8B01000000 <1> mov ebx, 1
35 34 00000022 B804000000 <1> mov eax, 4
36 35 00000027 CD00 <1> int 80h
37 36 <1>
38 37 00000029 5B <1> pop ebx
39 38 0000002A 59 <1> pop ecx
40 39 0000002B 5A <1> pop edx
41 40 0000002C C3 <1> ret
42 41 <1>
43 42 <1>
44 43 <1> ;----- sprintfLF -----
45 44 <1> ; Функция печати сообщения с переводом строки
46 45 <1> ; входные данные: mov eax, <message>
47 46 <1> sprintfLF:
48 47 0000002D EB0FFFFFFF <1> call sprintf
49 48 <1>
50 49 00000032 50 <1> push eax
51 50 00000033 B80A000000 <1> mov eax, 0Ah
52 51 00000038 50 <1> push eax
53 52 00000039 89E0 <1> mov eax, esp
54 53 0000003B ECFFFFFFF <1> call sprintf
55 54 00000040 58 <1> pop eax
56 55 00000041 58 <1> pop eax
57 56 00000042 C3 <1> ret
58 57 <1>
59 58 <1> ;----- sread -----
60 59 <1> ; Функция считывания сообщения
61 60 <1> ; входные данные: mov eax, <buffer>, mov ebx, <N>
62 61 <1> sread:
63 62 00000043 53 <1> push ebx
64 63 00000044 50 <1> push eax

```

Рис. 3.12: Изучение файла листинга

В представленных трех строчках содержаться следующие данные: ((рис. 3.13)

3	2	<1> ; Функция вычисления длины сообщения
4	3	<1> slen:
5	4 00000000 53	<1> push ebx

Рис. 3.13: Выбранные строки файла

“2” - номер строки кода, “; Функция вычисления длинны сообщения” - комментарий к коду, не имеет адреса и машинного кода.

“3” - номер строки кода, “slen” - название функции, не имеет адреса и машинного кода.

“4” - номер строки кода, “00000000” - адрес строки, “53” - машинный код, “push ebx” - исходный текст программы, инструкция “push” помещает операнд “ebx” в стек.

Открываю файл с программой lab7-2.asm и в выбранной мной инструкции с двумя операндами удаляю выделенный операнд. (рис. 3.14)

```

22 case test, вывод подпрограммы перевода символа в int
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx; 'ecx = A'
26 mov [max],ecx ; 'max = A'

```

Рис. 3.14: Удаление выделенного операнда из кода

Выполняю трансляцию с получением файла листинга. (рис. 3.15)

```

aanovikova123@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:25: error: invalid combination of opcode and operands

```

Рис. 3.15: Получение файла листинга

Терминал выводит сообщение об ошибке: инструкция mov не может работать, имея только один операнд, из-за чего нарушается работа кода.

Ознакамливаюсь с новым файлом листинга. В файле в этой строчке ошибка. (рис. 3.16)

```

199 24 ; ----- Записываем 'A' в переменную 'max'
200 25 mov ecx; 'ecx = A'
201 25 *****
202 26 00000110 890D[00000000] mov [max],ecx ; 'max = A'

```

Рис. 3.16: Ошибка в файле листинга

3.3 Задания для самостоятельной работы

Вариант 20

Задание 1

Пишу программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбираю из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Мой вариант под номером 20, поэтому мои значения - 95, 2, 61. (рис. 3.17)

```

Открыть  [иконка] *task-1.asm
~work/arch-pc/lab07

1 %include 'in_out.asm'
2
3 section .data
4     msg db "Наименьшее число: ", 0
5     A dd 95
6     B dd 2
7     C dd 61
8
9 section .bss
10    min resd 1 ; Переменная для хранения наименьшего значения (32-битная)
11
12 section .text
13 global _start
14
15 _start:
16     ; Инициализируем min значением A
17     mov eax, [A] ; Загружаем A в eax
18     mov [min], eax ; Сохраняем A в min
19
20     ; Сравниваем min и B
21     mov ebx, [B] ; Загружаем B в ebx
22     cmp eax, ebx ; Сравниваем min (eax) и B
23     jle _compare_c ; Если min <= B, идем к сравнению с C
24     mov [min], ebx ; Иначе B становится наименьшим
25
26 _compare_c:
27     ; Сравниваем min и C
28     mov ecx, [C] ; Загружаем C в ecx
29     mov eax, [min] ; Загружаем min в eax
30     cmp eax, ecx ; Сравниваем min и C
31     jle _end ; Если min <= C, завершаем
32     mov [min], ecx ; Иначе C становится наименьшим
33
34 _end:
35     ; Выводим сообщение
36     mov eax, msg
37     call sprint
38
39     ; Выводим значение min
40     mov eax, [min] ; Загружаем значение min в eax
41     call printf
42
43     ; Завершаем программу
44     call quit

```

Рис. 3.17: Написание программы

Создаю исполняемый файл и проверяю его работу, подставляя необходимые значение. (рис. 3.18)

```

aanovikova123@fedora:~/work/arch-pc/lab07$ nasm -f elf task-1.asm
aanovikova123@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 task-1.o -o task-1
aanovikova123@fedora:~/work/arch-pc/lab07$ ./task-1
Наименьшее число: 2
aanovikova123@fedora:~/work/arch-pc/lab07$

```

Рис. 3.18: Запуск файла и проверка его работы

Программа работает корректно.

Код программы:

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

```
section .data
```

```
msg db "Наименьшее число: ", 0
```

```
A dd 95
```

```
B dd 2
```

```
C dd 61
```

```
section .bss
```

```
    min resd 1 ; Переменная для хранения наименьшего значения (32-битная)
```

```
section .text
```

```
global _start
```

```
_start:
```

```
    ; Инициализируем min значением A
```

```
    mov eax, [A] ; Загружаем A в eax
```

```
    mov [min], eax ; Сохраняем A в min
```

```
    ; Сравниваем min и B
```

```
    mov ebx, [B] ; Загружаем B в ebx
```

```
    cmp eax, ebx ; Сравниваем min (eax) и B
```

```
    jle _compare_c ; Если min <= B, идем к сравнению с C
```

```
    mov [min], ebx ; Иначе B становится наименьшим
```

```
_compare_c:
```

```
    ; Сравниваем min и C
```

```
    mov ecx, [C] ; Загружаем C в ecx
```

```
    mov eax, [min] ; Загружаем min в eax
```

```
    cmp eax, ecx ; Сравниваем min и C
```

```
    jle _end ; Если min <= C, завершаем
```

```
    mov [min], ecx ; Иначе C становится наименьшим
```

```
_end:
```

```
    ; Выводим сообщение
```

```
    mov eax, msg
```

```
    call sprint
```

; Выводим значение min

mov **eax**, [**min**] *; Загружаем значение min в eax*

call **iprintLF**

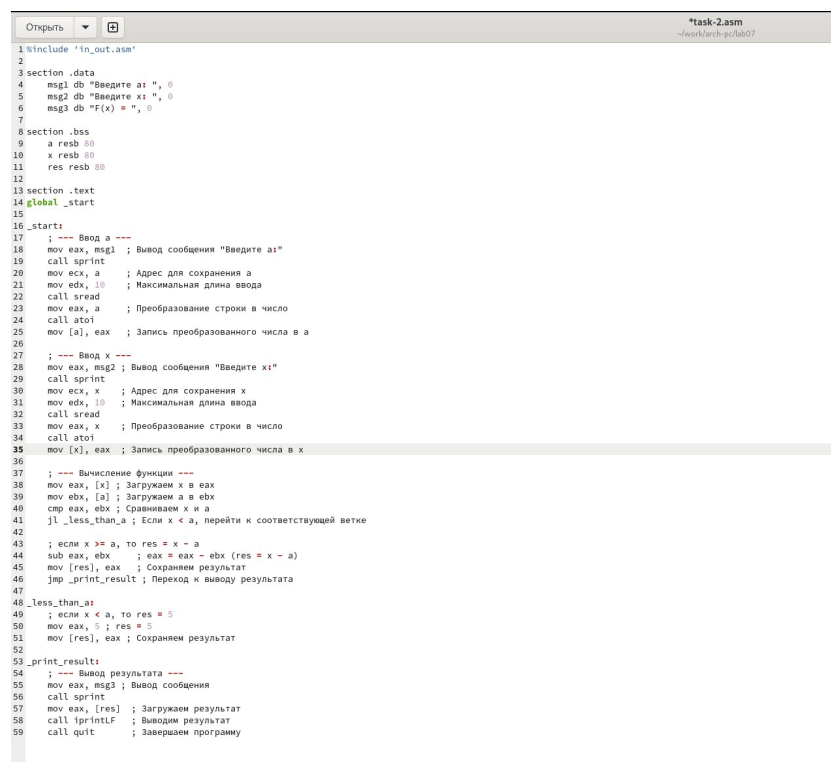
; Завершаем программу

call **quit**

Задание 2

Пишу программу (рис. 3.19), которая для введенных с клавиатуры значений x и a вычисляет значение и выводит результат вычислений заданной для моего варианта функции $f(x)$: $x - a$, если $x \geq a$;

5 , если $x < a$.



```
1 %include 'in_out.asm'
2
3 section .data
4     msg1 db "Введите a: ", 0
5     msg2 db "Введите x: ", 0
6     msg3 db "F(x) = ", 0
7
8 section .bss
9     a resb 80
10    x resb 80
11    res resb 80
12
13 section .text
14 global _start
15
16 _start:
17     ; --- Ввод a ---
18     mov eax, msg1 ; Вывод сообщения "Введите a:"
19     call sprint
20     mov ecx, a ; Адрес для сохранения a
21     mov edx, 10 ; Максимальная длина ввода
22     call sread
23     mov eax, a ; Преобразование строки в число
24     call atoi
25     mov [a], eax ; Запись преобразованного числа в a
26
27     ; --- Ввод x ---
28     mov eax, msg2 ; Вывод сообщения "Введите x:"
29     call sprint
30     mov ecx, x ; Адрес для сохранения x
31     mov edx, 10 ; Максимальная длина ввода
32     call sread
33     mov eax, x ; Преобразование строки в число
34     call atoi
35     mov [x], eax ; Запись преобразованного числа в x
36
37     ; --- Вычисление функции ---
38     mov eax, [x] ; Загружаем x в eax
39     mov ebx, [a] ; Загружаем a в ebx
40     cmp eax, ebx ; Сравниваем x и a
41     jl _less_than_a ; Если x < a, перейти к соответствующей ветке
42
43     ; если x >= a, то res = x - a
44     sub eax, ebx ; eax = eax - ebx (res = x - a)
45     mov [res], eax ; Сохраняем результат
46     jmp _print_result ; Переход к выводу результата
47
48 _less_than_a:
49     ; если x < a, то res = 5
50     mov eax, 5 ; res = 5
51     mov [res], eax ; Сохраняем результат
52
53 _print_result:
54     ; --- Вывод результата ---
55     mov eax, msg3 ; Вывод сообщения
56     call sprint
57     mov eax, [res] ; Загружаем результат
58     call iprintLF ; Выводим результат
59     call quit ; Завершаем программу
```

Рис. 3.19: Написание программы

Создаю исполняемый файл и проверяю его работу для значений x и a соответственно: (1, 2), (2, 1). (рис. 3.20)

```

aanovikova123@fedora:~/work/arch-pc/lab07$ nasm -f elf task-2.asm
aanovikova123@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 task-2.o -o task-2
aanovikova123@fedora:~/work/arch-pc/lab07$ ./task-2
Введите a: 2
Введите x: 1
F(x) = 5
aanovikova123@fedora:~/work/arch-pc/lab07$ ./task-2
Введите a: 1
Введите x: 2
F(x) = 1

```

Рис. 3.20: Запуск файла и проверка его работы

Программа работает корректно.

Код программы:

```

#include 'in_out.asm'

section .data
    msg1 db "Введите a: ", 0
    msg2 db "Введите x: ", 0
    msg3 db "F(x) = ", 0

section .bss
    a resb 80
    x resb 80
    res resb 80

section .text
global _start

_start:
    ; --- Ввод a ---
    mov eax, msg1 ; Вывод сообщения "Введите a:"
    call sprint
    mov ecx, a     ; Адрес для сохранения a
    mov edx, 10    ; Максимальная длина ввода

```



```

call sread
mov eax, a      ; Преобразование строки в число
call atoi
mov [a], eax    ; Запись преобразованного числа в a

; --- Ввод x ---
mov eax, msg2 ; Вывод сообщения "Введите x:"
call sprint
mov ecx, x      ; Адрес для сохранения x
mov edx, 10      ; Максимальная длина ввода
call sread
mov eax, x      ; Преобразование строки в число
call atoi
mov [x], eax    ; Запись преобразованного числа в x

; --- Вычисление функции ---
mov eax, [x] ; Загружаем x в eax
mov ebx, [a] ; Загружаем a в ebx
cmp eax, ebx ; Сравниваем x и a
jnl _less_than_a ; Если  $x < a$ , перейти к соответствующей ветке

; если  $x \geq a$ , то  $res = x - a$ 
sub eax, ebx    ;  $eax = eax - ebx$  ( $res = x - a$ )
mov [res], eax ; Сохраняем результат
jmp _print_result ; Переход к выводу результата

_less_than_a:
; если  $x < a$ , то  $res = 5$ 
mov eax, 5 ;  $res = 5$ 

```

```
mov [res], eax ; Сохраняем результат
```

```
_print_result:
```

```
    ; --- Вывод результата ---
```

```
mov eax, msg3 ; Вывод сообщения
```

```
call sprint
```

```
mov eax, [res] ; Загружаем результат
```

```
call iprintLF ; Выводим результат
```

```
call quit      ; Завершаем программу
```

4 Выводы

По итогам данной лабораторной работы были изучены команды условного и безусловного переходов, приобретены навыки написания программ с использованием переходов, а также было знакомство с назначением и структурой файла листинга.