

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ

### ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

дисциплина: Архитектура компьютера

Студент: Ким Илья

Группа: НКАбд-03-25

МОСКВА

2025 г.

## Содержание

1. Цель работы.....	4
2. Задание.....	5
3. Теоретическое введение.....	6
4. Выполнение лабораторной работы.....	7
4.1 Реализация переходов в NASM . . . . .	7
4.2 Изучение структуры файла листинга . . . . .	11
4.3 Задания для самостоятельной работы . . . . .	12
5. Выводы.....	21
6. Список литературы.....	22

## Список иллюстраций

4.1. Создание каталога и файла для программы . . . . .	7
4.2. Сохранение программы . . . . .	7
4.3. Запуск программы . . . . .	8
4.4. Изменение программы . . . . .	8
4.5. Запуск изменённой программы. . . . .	8
4.6. Изменение программы . . . . .	9
4.7. Проверка изменений. . . . .	9
4.8. Сохранение новой программы . . . . .	10
4.9. Проверка программы из листинга . . . . .	11
4.10. Проверка файла листинга . . . . .	12
4.11. Удаление операнда из программы. . . . .	13
4.12. Просмотр ошибки в файле листинга. . . . .	14
4.13. Первая программа самостоятельной работы. . . . .	15
4.14. Проверка работы первой программы . . . . .	17
4.15. Вторая программа самостоятельной работы. . . . .	18
4.16. Проверка работы второй программы . . . . .	20

# **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 файлов листинга.
3. Самостоятельное написание программ по материалам лабораторной работы.

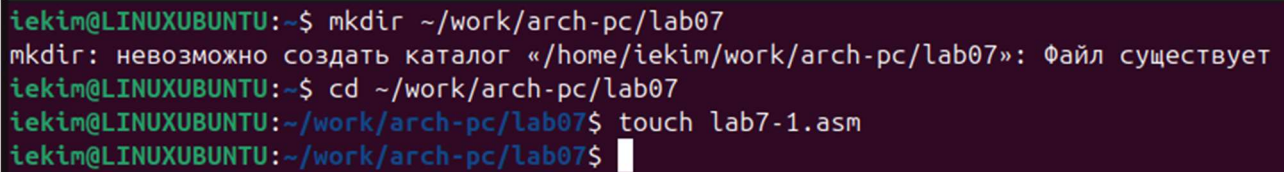
### **3. Теоретическое введение**

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов: условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия, без условного перехода – выполнение передачи управления в определенную точку программы без каких-либо условий.

## 4. Выполнение лабораторной работы

### 4.1. Реализация переходов в NASM

Создаю каталог для программ лабораторной работы №7 (рис.4.1).



```
iekim@LINUXUBUNTU:~$ mkdir ~/work/arch-pc/lab07
mkdir: невозможно создать каталог «/home/iekim/work/arch-pc/lab07»: Файл существует
iekim@LINUXUBUNTU:~$ cd ~/work/arch-pc/lab07
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$ touch lab7-1.asm
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$
```

Рис. 4.1: Создание каталога и файла для программы

Копирую код из листинга в файл будущей программы. (рис.4.2).

```
%include 'in_out.asm'

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
call sprintLF

_label2:
mov eax, msg2
call sprintLF

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 4.2: Сохранение программы

При запуске программы я убедился в том, что не условный переход действительно изменяет порядок выполнения инструкций (рис.4.3).

```

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

```

Рис. 4.3: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения функций (рис.4.4).

```

#include 'in_out.asm'

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
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF

_end:
call quit

```

Рис. 4.4: Изменение программы

Запускаю программу и проверяю, что примененные изменения верны (рис. 4.5).

```

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

```

Рис. 4.5: Запуск изменённой программы



Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис.4.6).

```
%include 'in_out.asm'

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
call sprintLF
jmp _end

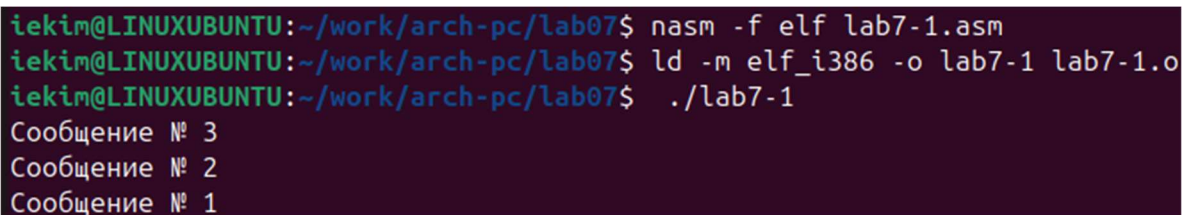
_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit
```

Рис. 4.6: Изменение программы

Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис.4.7).



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

Рис. 4.7: Проверка изменений

Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. 4.8).

```
%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:

mov eax, msg1
call sprint

mov ecx, B
mov edx, 10
call sread

mov eax, B
call atoi
mov [B], eax

mov ecx, [A]
mov [max], ecx

cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [max], ecx

check_B:
mov eax, max

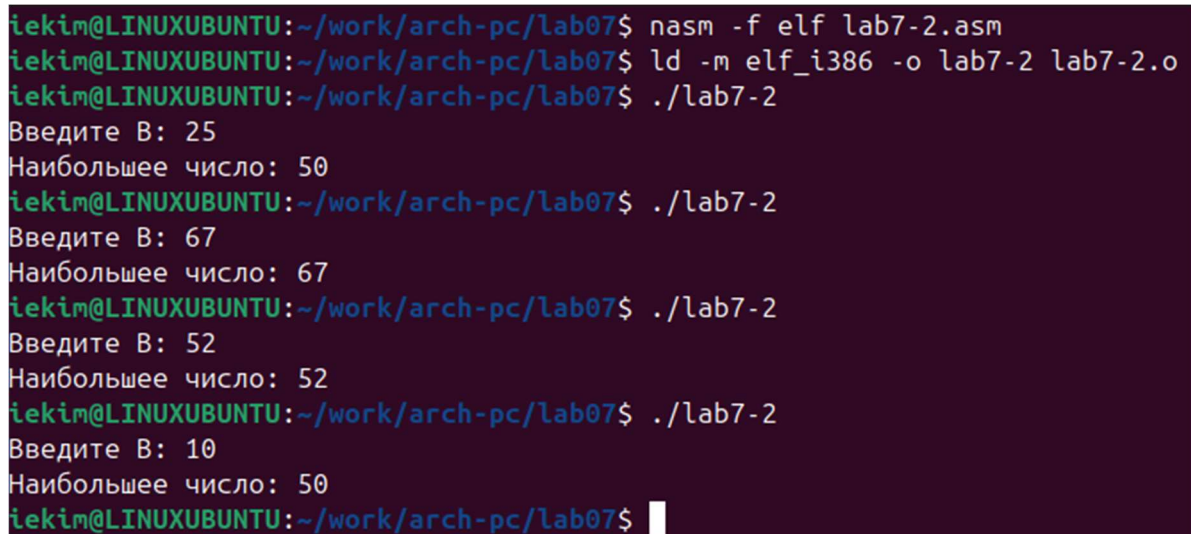
call atoi
mov [max], eax

mov ecx, [max]
cmp ecx, [B]
jg fin
mov ecx, [B]
mov [max], ecx

fin:
mov eax, msg2
call sprint
mov eax, [max]
call iprintLF
call quit
```

Рис. 4.8: Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными (рис.4.9).



```
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 25
Наибольшее число: 50
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 67
Наибольшее число: 67
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 52
Наибольшее число: 52
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 10
Наибольшее число: 50
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$
```

Рис. 4.9: Проверка программы из листинга

## 4.2 Изучение структуры файла листинга

Создаю файл листинга с помощью флага-l команды nasm и открываю его с помощью текстового редактора mousepad (рис.4.10).

```
1                               %include 'in_out.asm'
1                               <1> ;----- slen -----
2                               <1> ; Функция вычисления длины сообщения
3                               <1> slen:
4 00000000 53                   <1> push    ebx
5 00000001 89C3                 <1> mov     ebx, eax
6                               <1>
7                               <1> nextchar:
8 00000003 803800               <1> cmp     byte [eax], 0
9 00000006 7403                 <1> jz      finished
10 00000008 40                  <1> inc     eax
11 00000009 EBF8                <1> jmp     nextchar
12                               <1>
13                               <1> finished:
14 0000000B 29D8                <1> sub     eax, ebx
15 0000000D 5B                  <1> pop     ebx
16 0000000E C3                  <1> ret
17                               <1>
18                               <1>
19                               <1> ;----- sprint -----
20                               <1> ; Функция печати сообщения
21                               <1> ; входные данные: mov eax,<message>
22                               <1> sprint:
23 0000000F 52                   <1> push    edx
24 00000010 51                   <1> push    ecx
25 00000011 53                   <1> push    ebx
26 00000012 50                   <1> push    eax
27 00000013 E8E8FFFFFF          <1> call    slen
28                               <1>
29 00000018 89C2                <1> mov     edx, eax
30 0000001A 58                   <1> pop     eax
31                               <1>
32 0000001B 89C1                <1> mov     ecx, eax
33 0000001D B801000000          <1> mov     ebx, 1
34 00000022 B804000000          <1> mov     eax, 4
35 00000027 CD80                <1> int     80h
36                               <1>
37 00000029 5B                   <1> pop     ebx
38 0000002A 59                   <1> pop     ecx
39 0000002B 5A                   <1> pop     edx
40 0000002C C3                  <1> ret
41                               <1>
42                               <1>
43                               <1> ;----- sprintf -----
44                               <1> ; Функция печати сообщения с переводом строки
45                               <1> ; входные данные: mov eax,<message>
46                               <1> sprintf:
47 0000002D E8DDFFFFFF          <1> call    sprint
48                               <1>
49 00000032 50                   <1> push    eax
50 00000033 B80A000000          <1> mov     eax, 0Ah
51 00000038 50                   <1> push    eax
52 00000039 89E0                <1> mov     eax, esp
53 0000003B E8CFFFFFFF          <1> call    sprintf
54 00000040 58                   <1> pop     eax
55 00000041 58                   <1> pop     eax
56 00000042 C3                  <1> ret
57                               <1>
```

Рис. 4.10: Проверка файла листинга

Первое значение в файле листинга – номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями. Удаляю один операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис.4.11).

```

#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:

mov eax, msg1
call sprint

mov ecx, B
mov edx, 10
call sread

mov eax, B
call atoi
mov [B], eax

mov ecx, [A]
mov [max], ecx

cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [max], ecx

check_B:
mov eax,
call atoi
mov [max], eax

mov ecx, [max]
cmp ecx, [B]
jg fin
mov ecx, [B]
mov [max], ecx

fin:
mov eax, msg2
call sprint
mov eax, [max]
call iprintLF
call quit

```

Рис. 4.11: Удаление операнда из программы

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. 4.12).

---

```

5 00000035 32300000      A dd '20'
6 00000039 35300000      C dd '50'
7
8                          SECTION .bss
9 00000000 <res Ah>       max resb 10
10 0000000A <res Ah>      B resb 10
11
12                          SECTION .text
13                          GLOBAL _start
14                          _start:
15
16 00000000 B8[00000000]   mov eax, msg1
17 00000005 E88A000000     call sprint
18
19 0000000A B9[0A000000]   mov ecx, B
20 0000000F BA0A000000     mov edx, 10
21 00000014 E8AF000000     call sread
22
23 00000019 B8[0A000000]   mov eax, B
24 0000001E E8FE000000     call atoi
25 00000023 A3[0A000000]   mov [B], eax
26
27 00000028 8B0D[35000000] mov ecx, [A]
28 0000002E 890D[00000000] mov [max], ecx
29
30 00000034 3B0D[39000000] cmp ecx, [C]
31 0000003A 7F0C          jg check_B
32 0000003C 8B0D[39000000] mov ecx, [C]
33 00000042 890D[00000000] mov [max], ecx
34
35                          check_B:
36                          mov eax,
36          *****      error: invalid combination of opcode and operands

```

Рис. 4.12: Просмотр ошибки в файле листинга

### 4.3 Задания для самостоятельной работы

Не понимаю, какой вариант я должен был получить вовремя 7 лабораторной работы, поэтому буду использовать свой вариант – девятый из предыдущей лабораторной работы. Возвращаю операнд к функции в программе и изменяю её так, чтобы она выводила переменную с наименьшим значением (рис. 4.13).

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '24'
C dd '15'

SECTION .bss
min resb 10
B resb 10

SECTION .text
GLOBAL _start
_start:

    mov eax, msg1
    call sprint

    mov ecx, B
    mov edx, 10
    call sread

    mov eax, B
    call atoi
    mov [B], eax

    mov ecx, [A]
    mov [min], ecx

    cmp ecx, [C]
    jg check_B
    mov ecx, [C]
    mov [min], ecx

check_B:
    mov eax, min
    call atoi
    mov [min], eax

    mov ecx, [min]
    cmp ecx, [B]
    jg fin
    mov ecx, [B]
    mov [min], ecx

fin:
    mov eax, msg2
    call sprint
    mov eax, [min]
    call iprintLF
    call quit
```

Рис. 4.13: Первая программа самостоятельной работы

Код первой программы:

```
%include 'in_out.asm'

SECTION .data

msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '24'
```

```

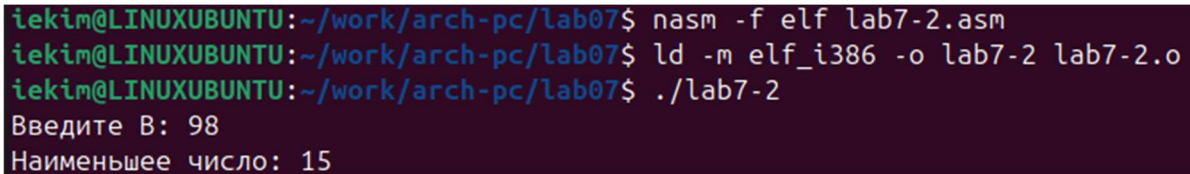
C dd '15'
SECTION .bss
min resb 10
B resb 10
SECTION .text
GLOBAL _start
_start:
mov eax, msg1
call sprint
mov ecx, B
mov edx, 10
call sread
mov eax, B
call atoi
mov [B], eax
mov ecx, [A]
mov [min], ecx
cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [min], ecx
check_B:
mov eax, min
call atoi
mov [min], eax
mov ecx, [min]
cmp ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx
fin:
mov eax, msg2

```



```
call sprint  
mov eax, [min]  
call iprintLF  
call quit
```

Проверяю корректность написания первой программы (рис.4.14).



```
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm  
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o  
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 98  
Наименьшее число: 15
```

Рис. 4.14: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатур переменных а и х (рис. 4.15).

```

#include "in_out.asm"
SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
rem: DB 'Результат: ', 0
SECTION .bss
x RESB 80
a RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax, msg_x
    call sprint
    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi
    mov edi, eax

    mov eax, msg_a
    call sprint
    mov ecx, a
    mov edx, 80
    call sread

    mov eax, a
    call atoi
    mov esi, eax

    cmp edi, esi
    jle add_values
    mov eax, edi
    jmp print_result

add_values:
    mov eax, edi
    add eax, esi

print_result:
    mov edi, eax
    mov eax, rem
    call sprint
    mov eax, edi
    call iprintLF
    call quit

```

Рис. 4.15: Вторая программа самостоятельной работы

Код второй программы:

```

#include 'in_out.asm'

SECTION .data

msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 0

SECTION .bss

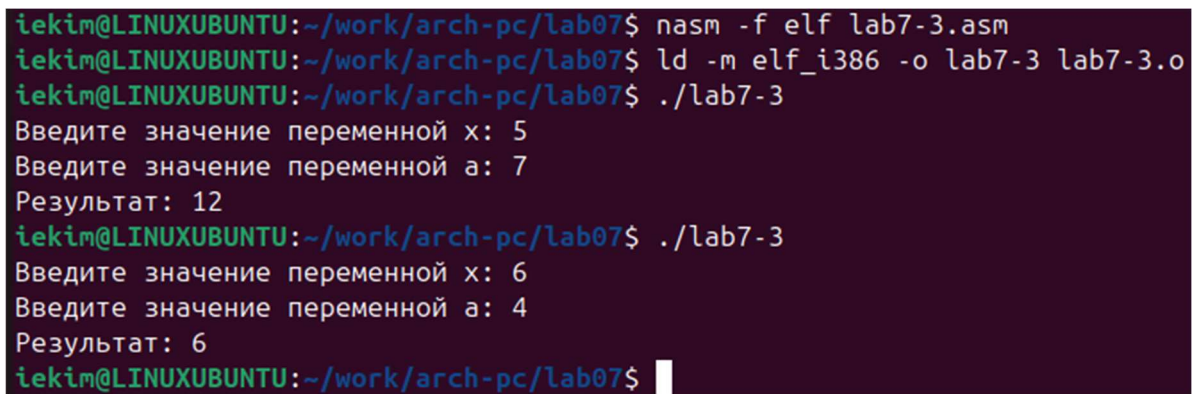
x: RESB 80
a: RESB 80

```

```
SECTION .text
GLOBAL _start
_start:
    mov eax, msg_x
    call sprint
mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi
    mov edi, eax
    mov eax, msg_a
    call sprint
    mov ecx, a
    mov edx, 80
    call sread
    mov eax, a
    call atoi
    mov esi, eax
    cmp edi, esi
    jle add_values
    mov eax, esi
    jmp print_result
add_values:
    mov eax, edi
    add eax, esi
print_result:
    mov edi, eax
    mov eax, res
    call sprint
mov eax, edi
    call iprintLF
```

call quit

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений а и х (рис.4.16).



```
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 12
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 6
iekim@LINUXUBUNTU:~/work/arch-pc/lab07$
```

Рис. 4.16: Проверка работы второй программы

## **5 Выводы**

При выполнении лабораторной работы я изучил команды условных и безусловных переходов, а также приобрёл навыки написания программ с использованием переходов, познакомился с назначением и структурой файлов листинга.

## **Список литературы**

1. Курс на ТУИС
2. Лабораторная работа №7
3. Программирование на языке ассемблера NASM Столяров. А. В.