

# 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: Гасанова Шакира Чингизовна

Группа: НКАбд-05-24

МОСКВА

2024 г.

# Содержание

<b>1 Цель работы .....</b>	<b>3</b>
<b>2 Задание .....</b>	<b>4</b>
<b>3 Теоретическое введение .....</b>	<b>5</b>
<b>4 Выполнение лабораторной работы</b>	
4.1 Реализация переходов в NASM.....	6
4.2 Изучение структуры файла листинга.....	11
4.3 Выполнение заданий для самостоятельной работы.....	15
<b>5 Выводы .....</b>	<b>22</b>
<b>6 Источники .....</b>	<b>23</b>

## **1 Цель работы**

Целью данной лабораторной работы является 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**

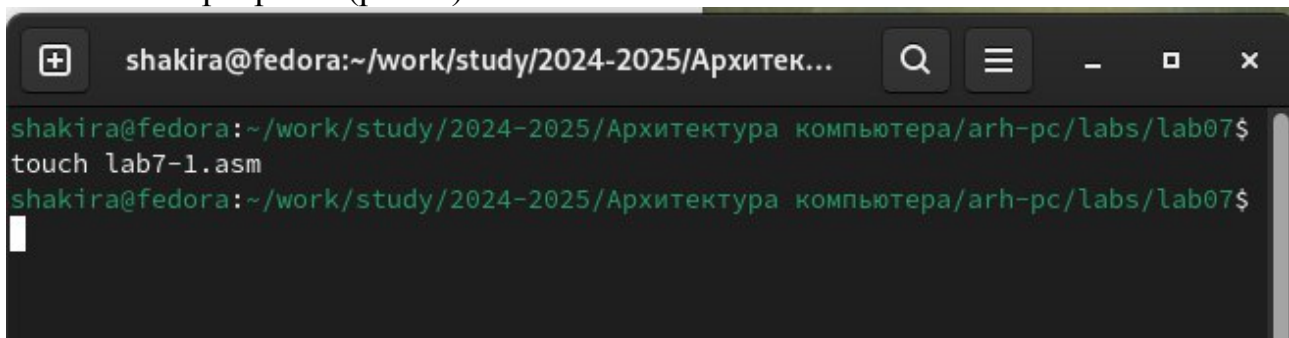
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

## 4 Выполнение лабораторной работы

### 4.1 Реализация переходов в NASM

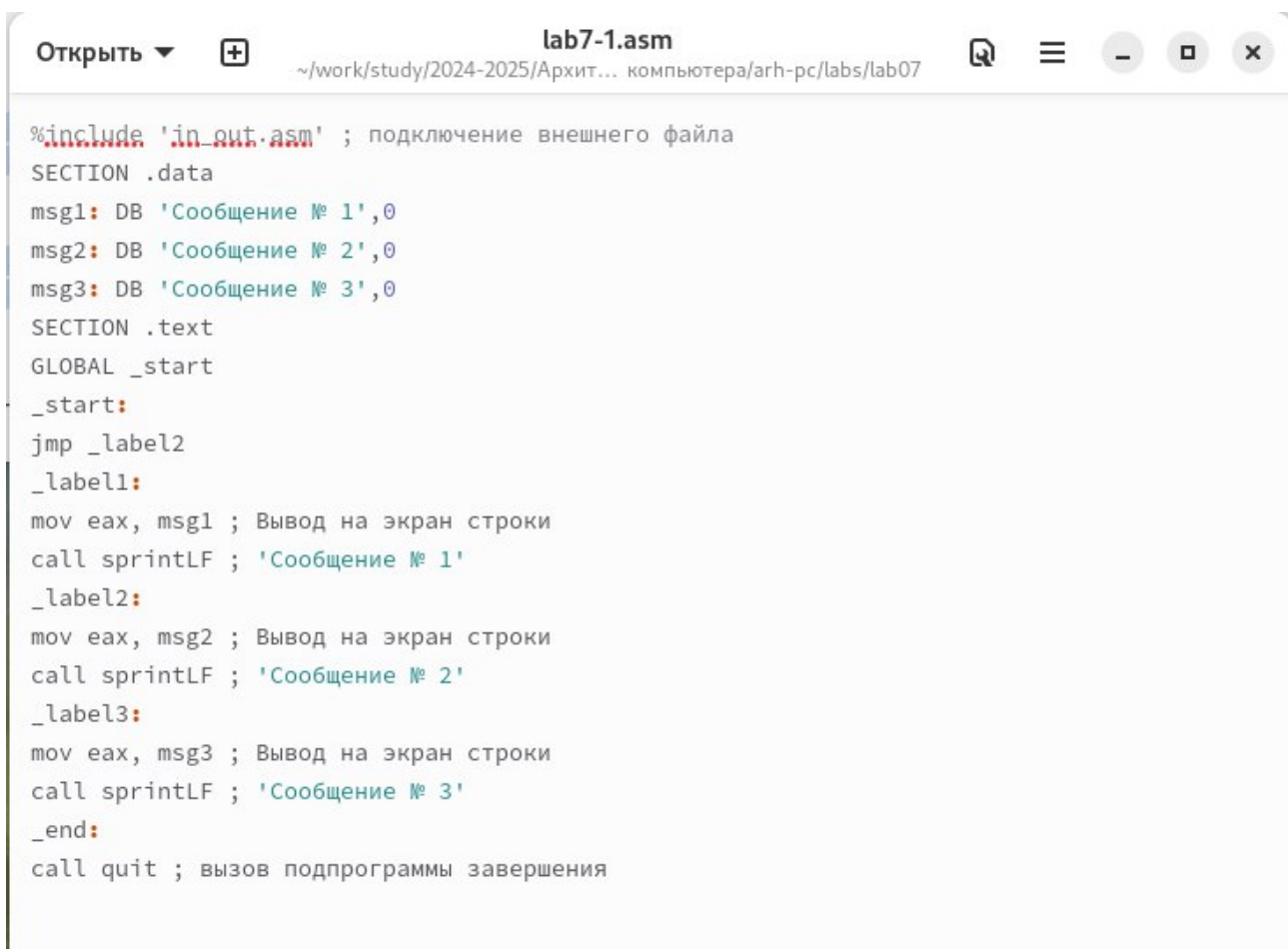
Создаю каталог для программ лабораторной работы №7 и файл для написания программ (рис. 1).



```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
touch lab7-1.asm  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$
```

Рис.1 Создание файла для программы

Копирую код из листинга в файл будущей программы (рис. 2)



```
Открыть ▾ lab7-1.asm  
~/work/study/2024-2025/Архит... компьютера/arh-pc/labs/lab07  
%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 Создание программы

При запуске программы я убедилась в том, что безусловный переход действительно изменяет порядок выполнения инструкций (рис. 3).

```

shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$
nasm -f elf lab7-1.asm
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$
ld -m elf_i386 -o lab7-1 lab7-1.o
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$
./lab7-1
Сообщение № 2
Сообщение № 3
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$

```

Рис.3 Запуск исполняемого файла

Изменяю программу таким образом, чтобы поменялся порядок выполнения функций (рис. 4).

```

lab7-1.asm
~/work/study/2024-2025/Архит... компьютера/arh-pc/labs/lab07

%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'
jump _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jump _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

```

Рис.4 Изменение программы

Запускаю программу и проверяю, что примененные изменения верны (рис. 5).

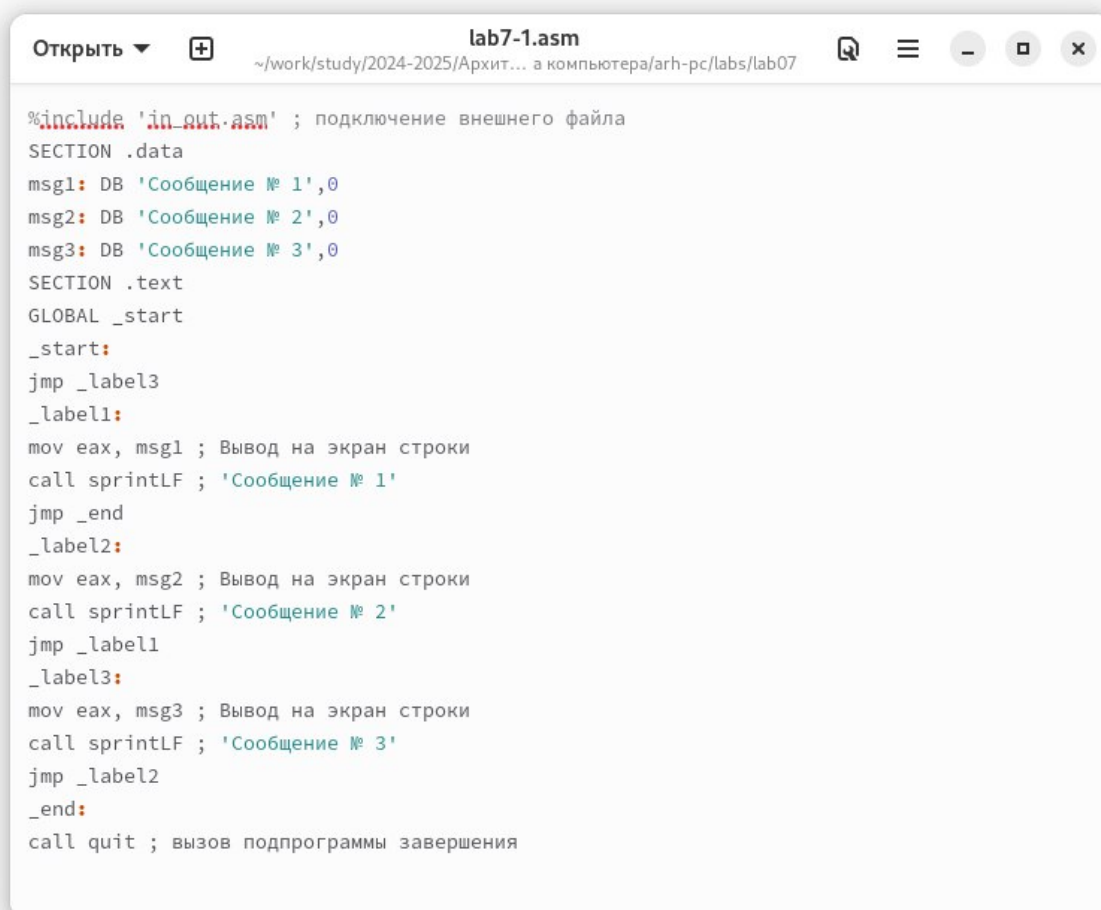
```

shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$
nasm -f elf lab7-1.asm
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$
ld -m elf_i386 -o lab7-1 lab7-1.o
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$
./lab7-1
Сообщение № 2
Сообщение № 1
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$

```

Рис.5 Запуск изменённой программы

Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. 6)



```

lab7-1.asm
~/work/study/2024-2025/Архит... а компьютера/arh-pc/labs/lab07

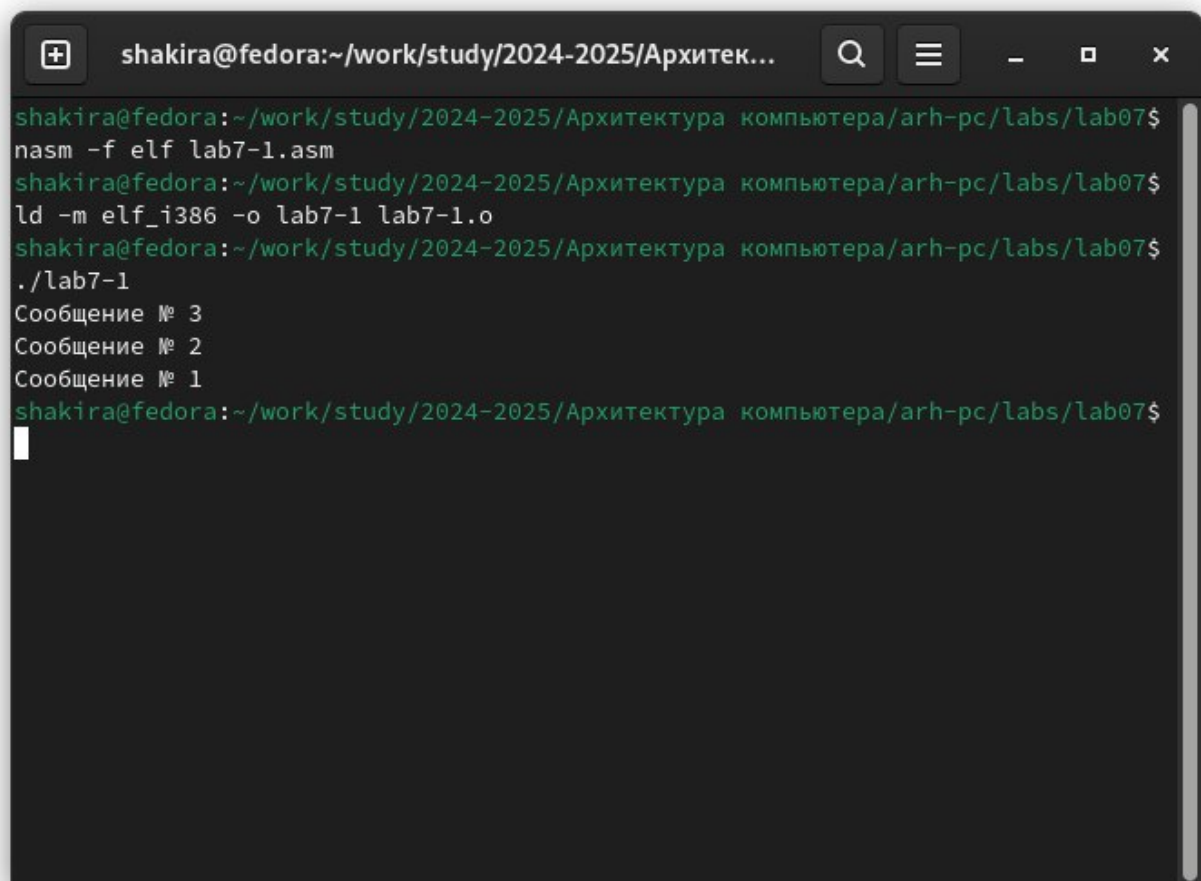
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; 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.6 Изменение программы

Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. 7)



A terminal window with a dark background and light green text. The window title is "shakira@fedora:~/work/study/2024-2025/Архитек...". The terminal shows the following commands and output:

```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
nasm -f elf lab7-1.asm  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
ld -m elf_i386 -o lab7-1 lab7-1.o  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
./lab7-1  
Сообщение № 3  
Сообщение № 2  
Сообщение № 1  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$
```

Рис.7 Проверка изменений

Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. 8).

```

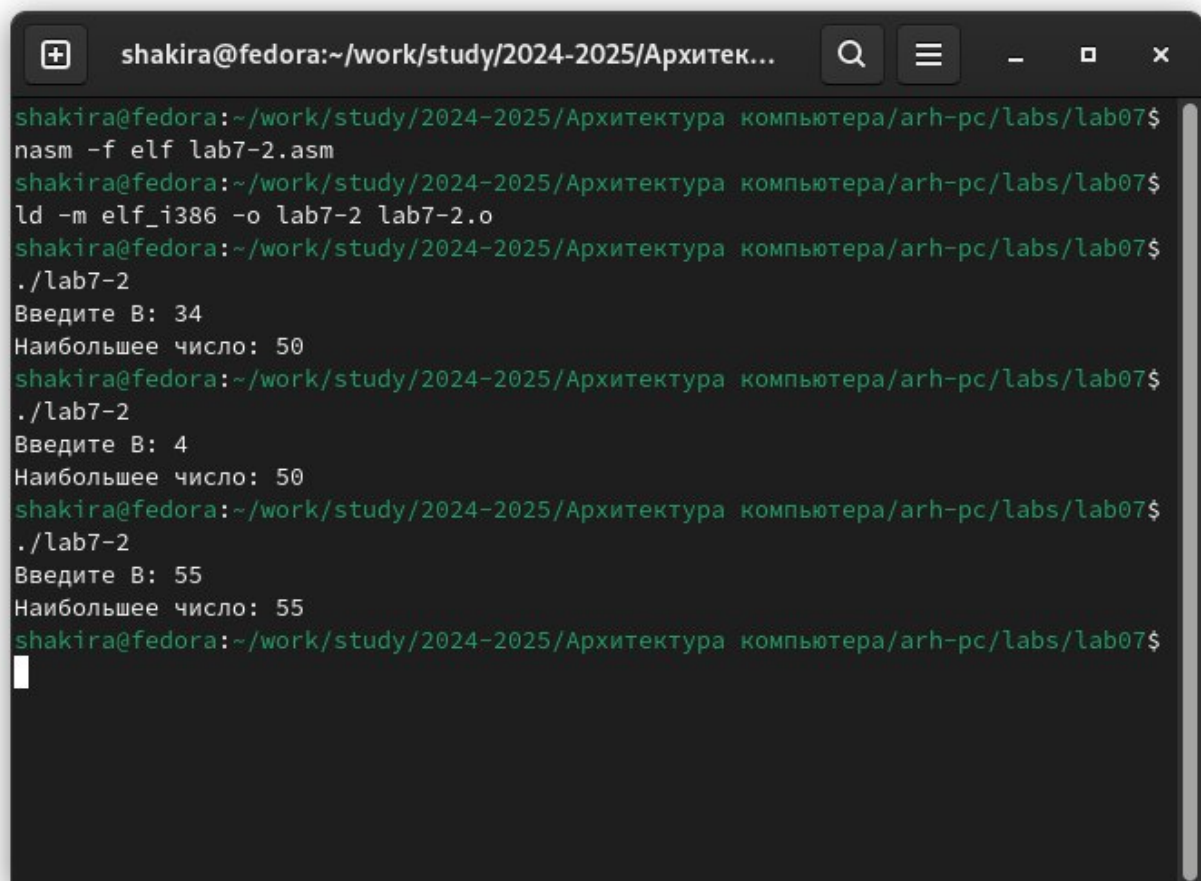
Открыть ▾  +  lab7-2.asm
~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07

%include 'io_out.asm'
section .data
msg1 db 'Введите A: ',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 A: '
mov eax,msg1
call sprint
; ----- Ввод 'A'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'A' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'A'
; ----- 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 iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход

```

Рис.8 Создание новой программы

Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными (рис. 9).

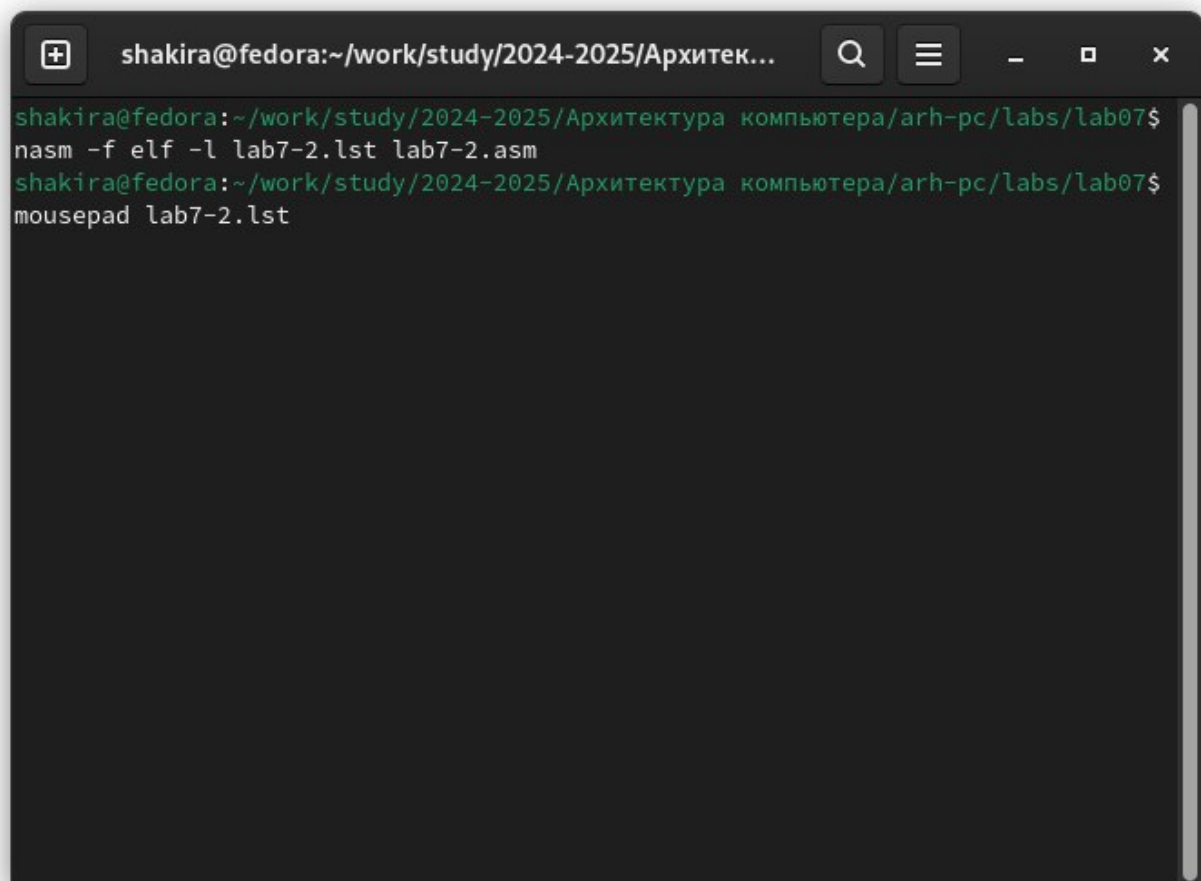


```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
nasm -f elf lab7-2.asm  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
ld -m elf_i386 -o lab7-2 lab7-2.o  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
./lab7-2  
Введите В: 34  
Наибольшее число: 50  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
./lab7-2  
Введите В: 4  
Наибольшее число: 50  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
./lab7-2  
Введите В: 55  
Наибольшее число: 55  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
█
```

Рис.9 Проверка программы

## 4.2 Изучение структуры файла листинга

Создаю файл листинга с помощью флага -l команды nasm и открываю его с помощью текстового редактора mousepad (рис. 10, 11).

A terminal window with a dark background and light green text. The window title is "shakira@fedora:~/work/study/2024-2025/Архитек...". The terminal shows three lines of commands: "nasm -f elf -l lab7-2.lst lab7-2.asm", "shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07\$", and "mousepad lab7-2.lst".

```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
nasm -f elf -l lab7-2.lst lab7-2.asm  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
mousepad lab7-2.lst
```

Рис.10 Создание файла

```
~\work\study\2024-2025\Архитектура компьютера\arh-pc\labs\lab07\lab7-2.lst - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

76      <1> iprint:
77 00000054 50      <1>      push    eax
78 00000055 51      <1>      push    ecx
79 00000056 52      <1>      push    edx
80 00000057 56      <1>      push    esi
81 00000058 B900000000 <1>      mov     ecx, 0
82      <1>
83      <1> divideLoop:
84 0000005D 41      <1>      inc     ecx
85 0000005E BA00000000 <1>      mov     edx, 0
86 00000063 BE0A000000 <1>      mov     esi, 10
87 00000068 F7FE      <1>      idiv    esi
88 0000006A 83C230      <1>      add     edx, 48
89 0000006D 52      <1>      push    edx
90 0000006E 83F800      <1>      cmp     eax, 0
91 00000071 75EA      <1>      jnz     divideLoop
92      <1>
93      <1> printLoop:
94 00000073 49      <1>      dec     ecx
95 00000074 89E0      <1>      mov     eax, esp
96 00000076 E894FFFFFF <1>      call    sprint
97 0000007B 58      <1>      pop     eax
98 0000007C 83F900      <1>      cmp     ecx, 0
99 0000007F 75F2      <1>      jnz     printLoop
100     <1>
101 00000081 5E      <1>      pop     esi
102 00000082 5A      <1>      pop     edx
103 00000083 59      <1>      pop     ecx
104 00000084 58      <1>      pop     eax
```

Рис.11 Проверка файла

Первое значение в файле листинга представляет собой номер строки, который может не совпадать с изначальным номером строки в исходном файле. Далее следует адрес — смещение машинного кода относительно начала текущего сегмента, затем сам машинный код. Завершает строку исходный текст программы с комментариями. Для проверки поведения файла листинга вношу изменения: удаляю один операнд из случайной инструкции (рис. 12).

```
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,|
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:
```

Рис.12 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. 13).

```

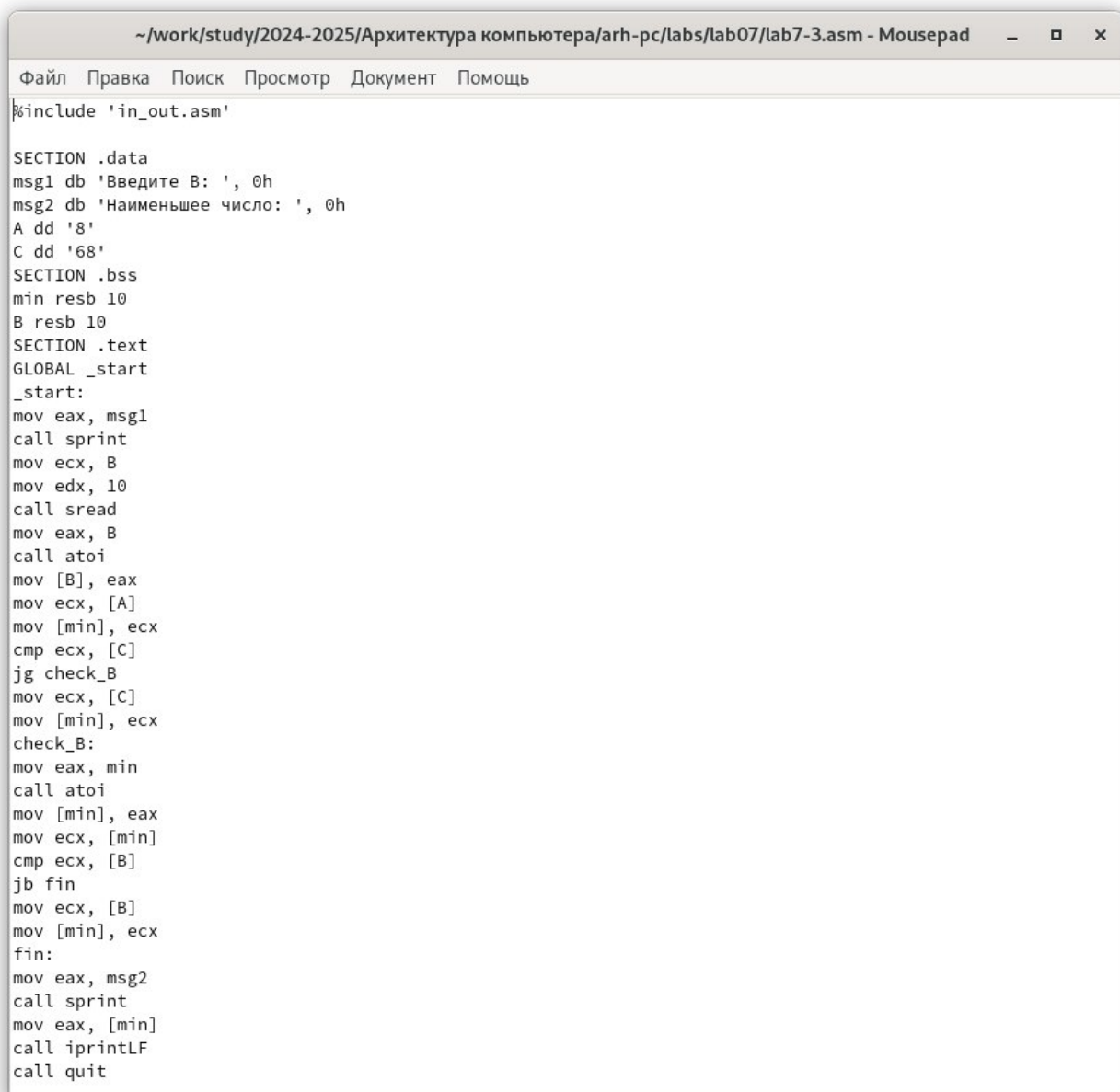
~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07/lab7-2.lst - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
5  00000035 32300000      A dd '20'
6  00000039 35300000      C dd '50'
7                                     section .bss
8  00000000 <res Ah>      max resb 10
9  0000000A <res Ah>      B resb 10
10                                    section .text
11      global _start
12      _start:
13          ; ----- Вывод сообщения 'Введите B: '
14  000000E8 B8[00000000]  mov eax,msg1
15  000000ED E81DFFFFFF    call sprint
16          ; ----- Ввод 'B'
17  000000F2 B9[0A000000]  mov ecx,B
18  000000F7 E847FFFFFF    mov edx,
19                                     error: invalid combination of opcode and operands
20      call sread
21          ; ----- Преобразование 'B' из символа в число
22  000000FC B8[0A000000]  mov eax,B
23  00000101 E896FFFFFF    call atoi ; Вызов подпрограммы перевода символа в число
24  00000106 A3[0A000000]  mov [B],eax ; запись преобразованного числа в 'B'
25          ; ----- Записываем 'A' в переменную 'max'
26  0000010B 8B0D[35000000]  mov ecx,[A] ; 'ecx = A'
27  00000111 890D[00000000]  mov [max],ecx ; 'max = A'
28          ; ----- Сравниваем 'A' и 'C' (как символы)
29  00000117 3B0D[39000000]  cmp ecx,[C] ; Сравниваем 'A' и 'C'
30  0000011D 7F0C      jg check_B ; если 'A>C', то переход на метку 'check_B',
31  0000011F 8B0D[39000000]  mov ecx,[C] ; иначе 'ecx = C'
32  00000125 890D[00000000]  mov [max],ecx ; 'max = C'
33          ; ----- Преобразование 'max(A,C)' из символа в число
34      check B:

```

Рис.13 Просмотр ошибки в файле листинга

### 4.3 Выполнение заданий для самостоятельной работы

Выполняю 4 вариант. Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 14).



```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '8'
C dd '68'
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
```

Рис.14 Написание программы

Код программы:

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '8'
C dd '68'
SECTION .bss
min resb 10
B resb 10
SECTION .text
```

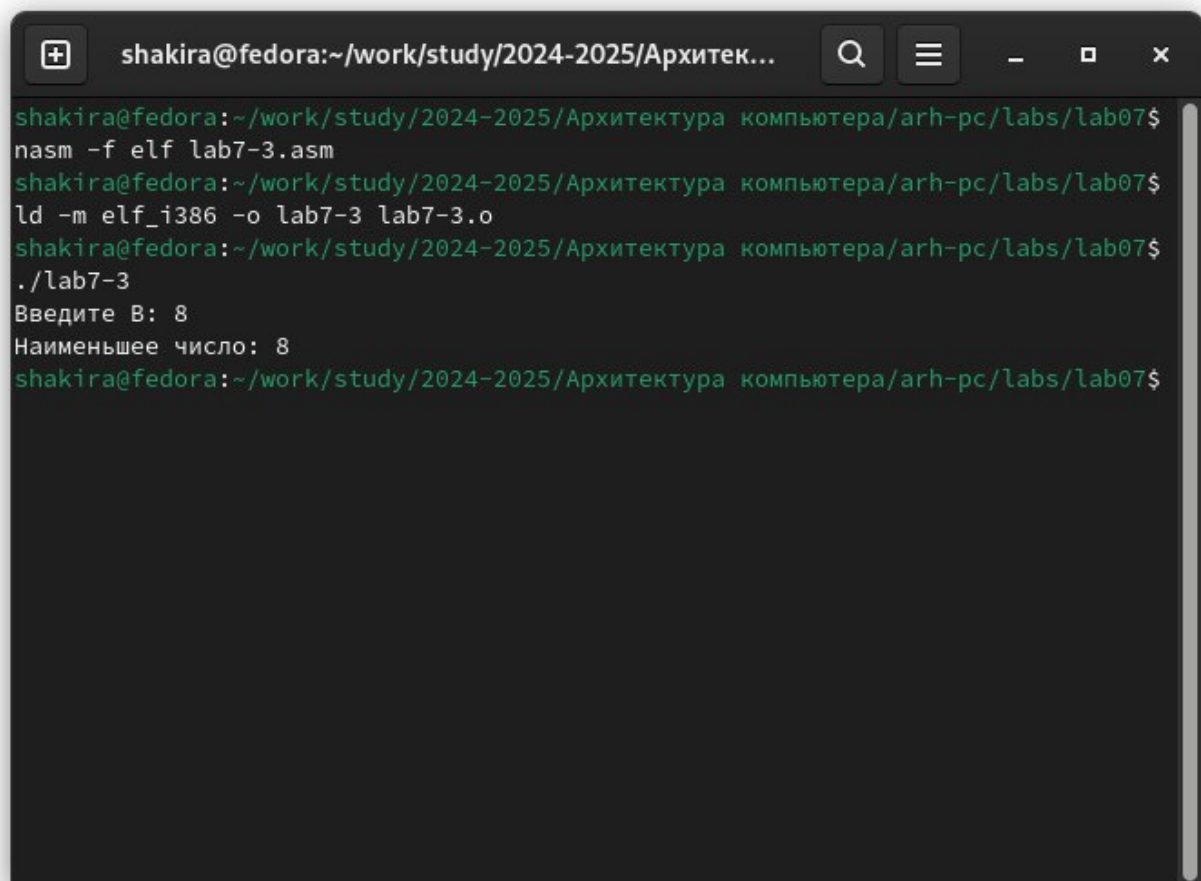


```

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

```

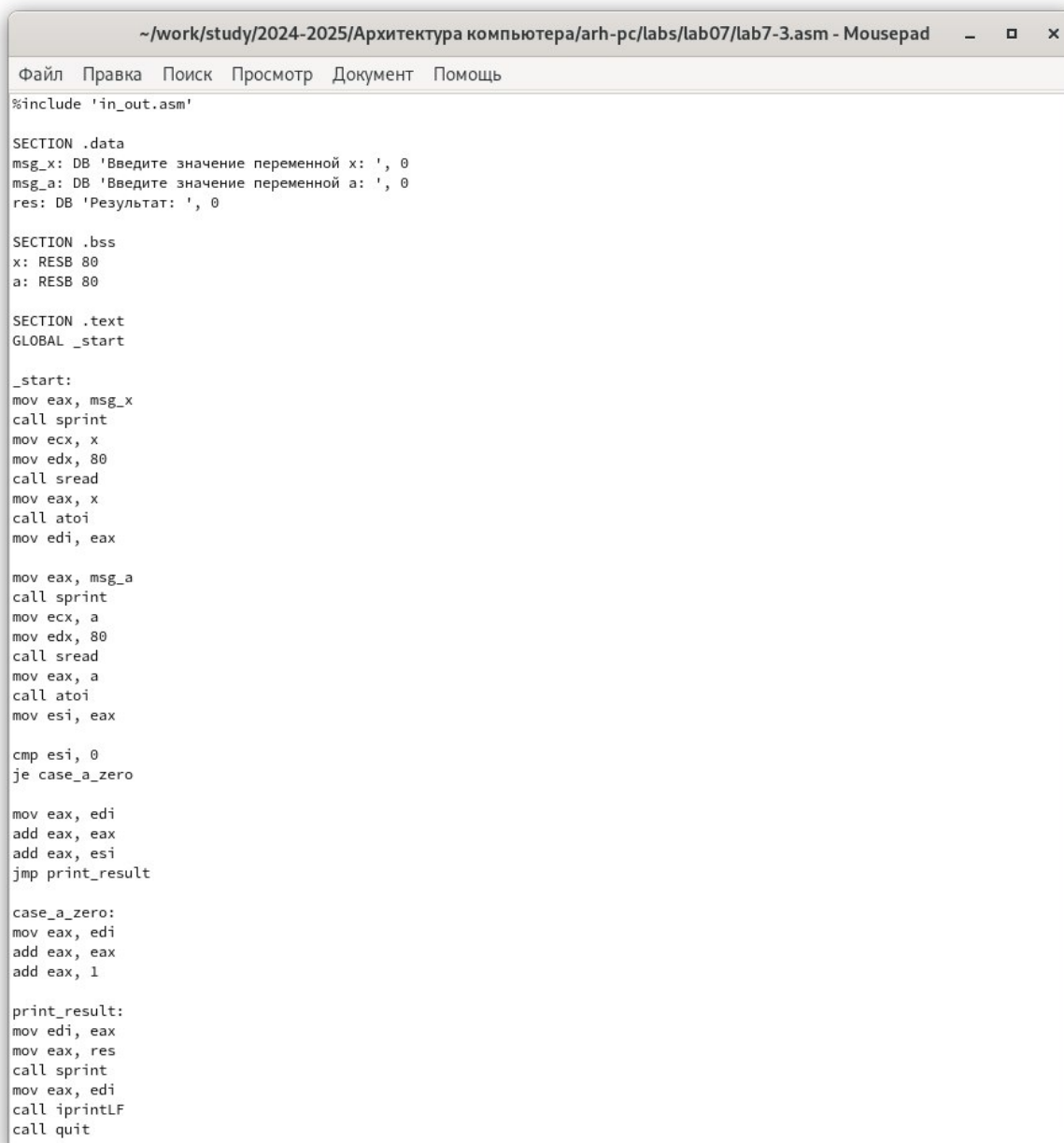
Проверяю корректность написания первой программы (рис. 15).



```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
nasm -f elf lab7-3.asm  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
ld -m elf_i386 -o lab7-3 lab7-3.o  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
./lab7-3  
Введите В: 8  
Наименьшее число: 8  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$
```

Рис.15 Запуск программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатуры переменных а и х (рис. 16).



```
~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07/lab7-3.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

%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 esi, 0
je case_a_zero

mov eax, edi
add eax, eax
add eax, esi
jmp print_result

case_a_zero:
mov eax, edi
add eax, eax
add eax, 1

print_result:
mov edi, eax
mov eax, res
call sprint
mov eax, edi
call iprintLF
call quit
```

Рис.16 Написание программы

Код 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
```

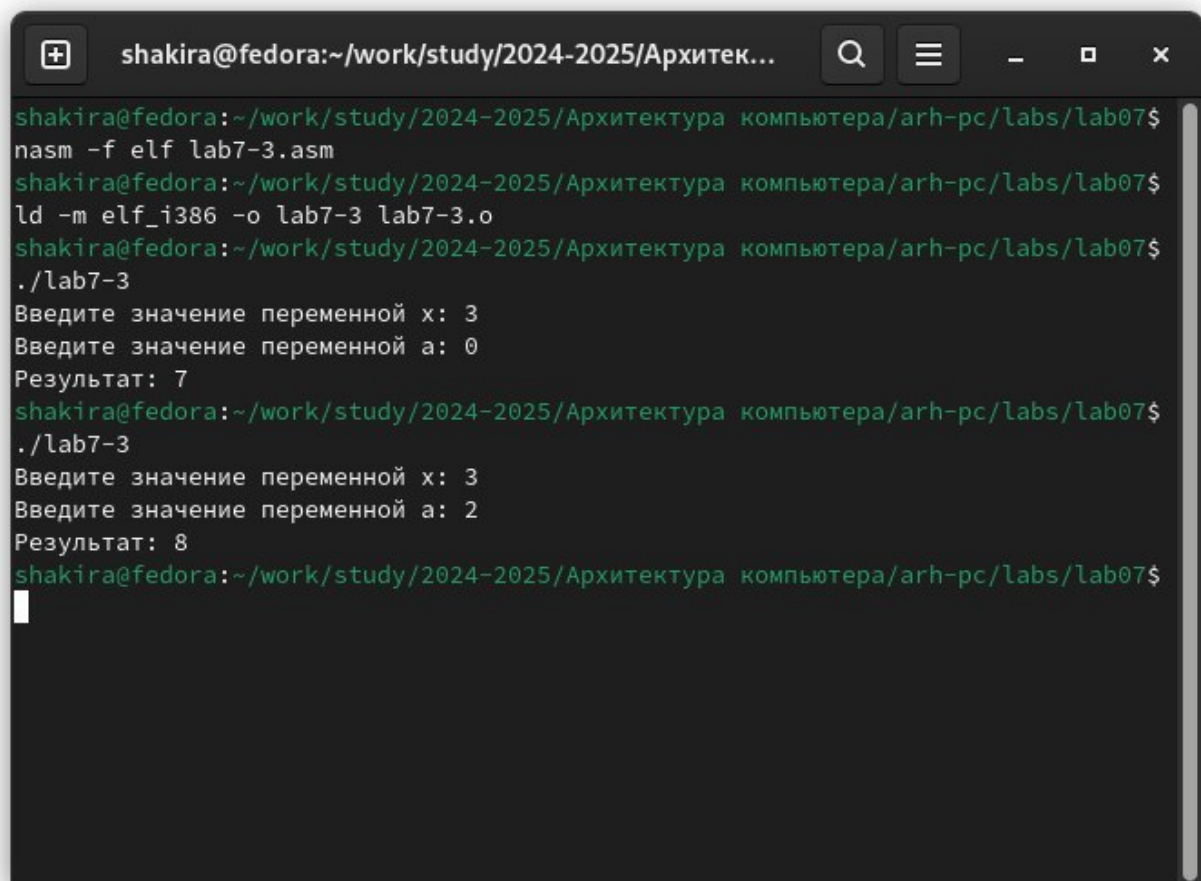
```
cmp esi, 0
je case_a_zero
```

```
mov eax, edi
add eax, eax
add eax, esi
jmp print_result
```

```
case_a_zero:
mov eax, edi
add eax, eax
add eax, 1
```

```
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. 17).



```
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
nasm -f elf lab7-3.asm  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
ld -m elf_i386 -o lab7-3 lab7-3.o  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
./lab7-3  
Введите значение переменной x: 3  
Введите значение переменной a: 0  
Результат: 7  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
./lab7-3  
Введите значение переменной x: 3  
Введите значение переменной a: 2  
Результат: 8  
shakira@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab07$  
█
```

Рис.17 Запуск программы

## **5 Выводы**

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

## **6 Источники**

1. [Архитектура ЭВМ \(rudn.ru\)](http://rudn.ru)