

# Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Хрисанова Ксения Олеговна

## Содержание

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

## 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. Выполнение заданий для самостоятельной работы

## 3 Теоретическое введение

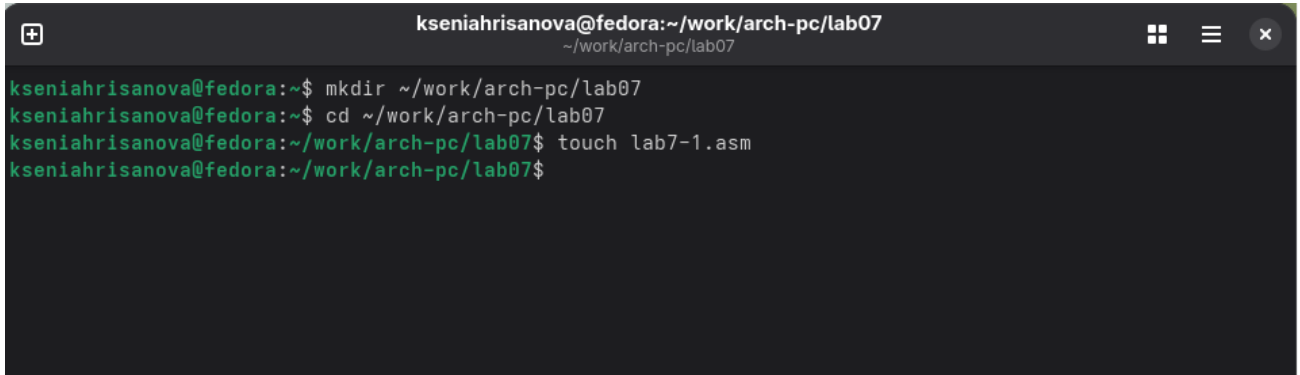
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- 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

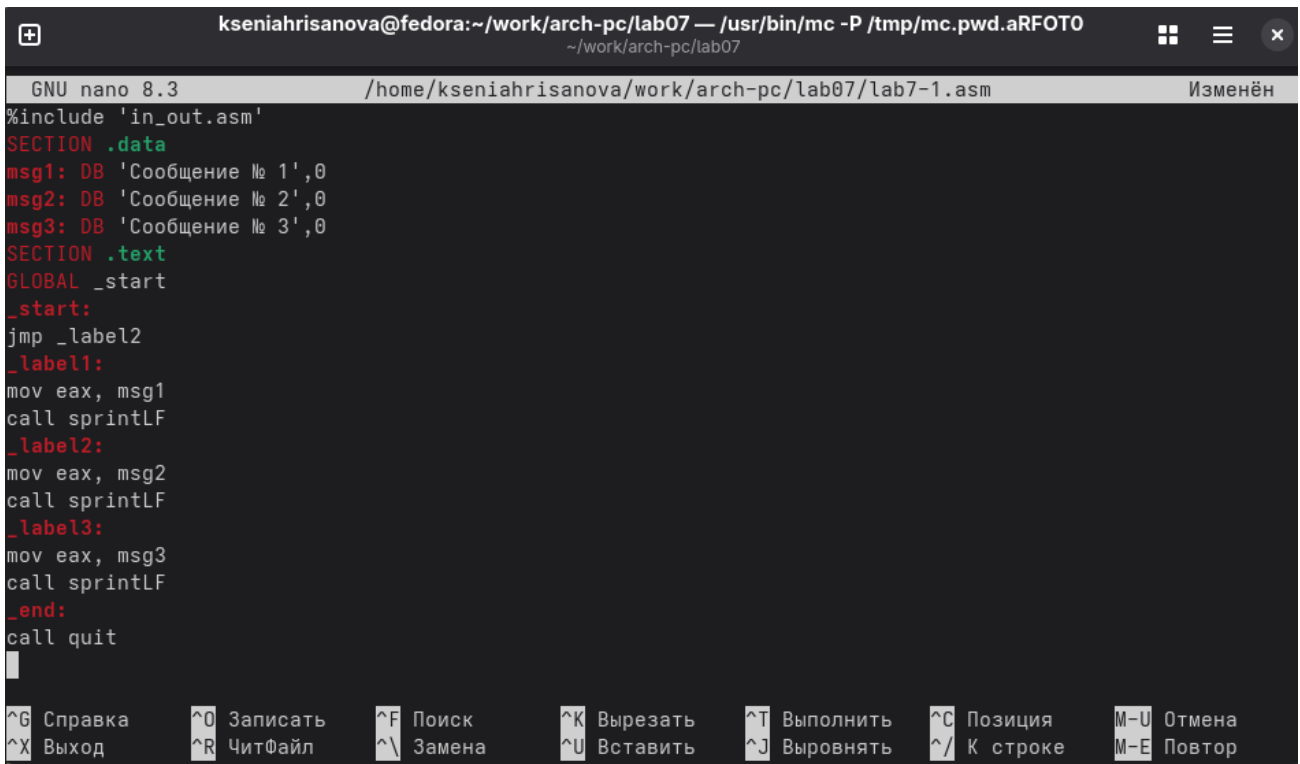
Создаю каталог для программ лабораторной работы №7 (рис. 1).



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

Рис. 1: Создание каталога и файла для программы

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



```
GNU nano 8.3 /home/kseniahrisnova/work/arch-pc/lab07/lab7-1.asm
%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. 2: Сохранение программы

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

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

Рис. 3: Запуск программы

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



```
GNU nano 8.3 /home/kseniahrisnova/work/arch-pc/lab07/lab7-1.asm
#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 sprintf
jmp _end
_label2:
mov eax, msg2
call sprintf
jmp _label1
_label3:
mov eax, msg3
call sprintf
_end:
call quit
```

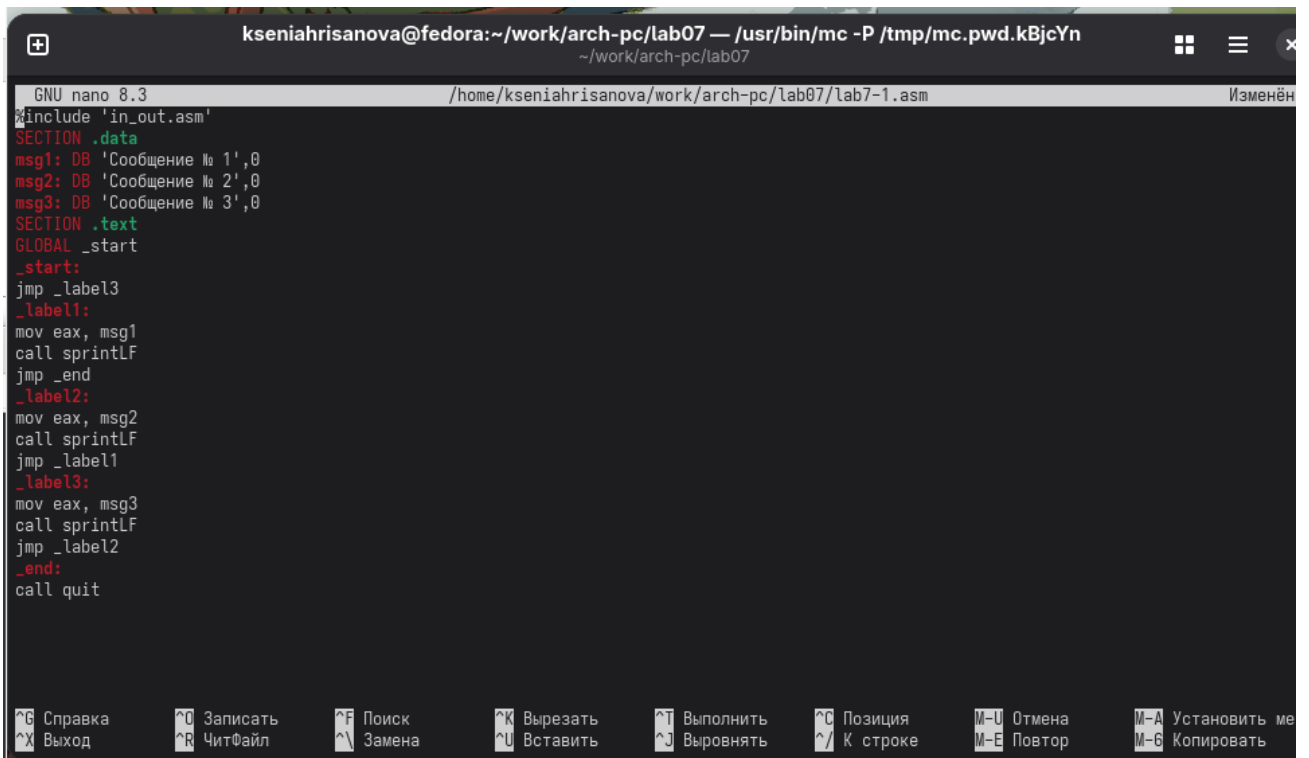
Рис. 4: Изменение программы

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

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

Рис. 5: Запуск измененной программы

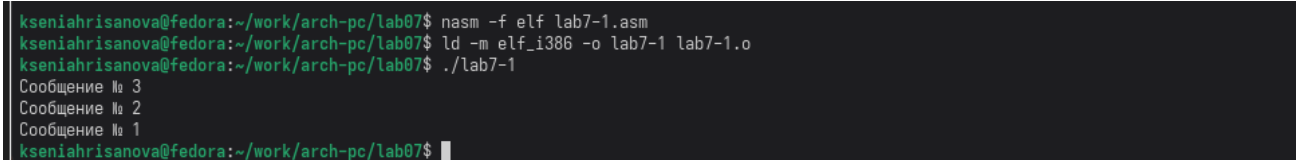
Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. 6).



```
kseniahrisnova@fedora:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.pwd.kBjcYn
~/work/arch-pc/lab07
GNU nano 8.3 /home/kseniahrisnova/work/arch-pc/lab07/lab7-1.asm
#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 sprintf
jmp _end
_label2:
mov eax, msg2
call sprintf
jmp _label1
_label3:
mov eax, msg3
call sprintf
jmp _label2
_end:
call quit
```

Рис. 6: Изменение программы

Программа работает корректно (рис. 7).



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

Рис. 7: Проверка изменений

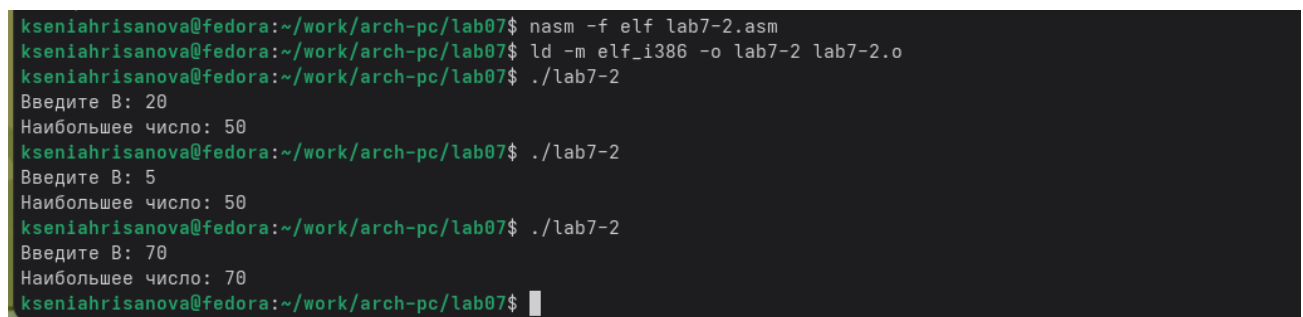
Создаю новый рабочий файл и вставляю в него код из листинга 7.3 (рис. 8).



```
GNU nano 8.3 /home/kseniahrisanova/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:
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 iprintf
call quit
```

Рис. 8: Сохранение новой программы

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



```
kseniahrisanova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
kseniahrisanova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
kseniahrisanova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 20
Наибольшее число: 50
kseniahrisanova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
kseniahrisanova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 70
Наибольшее число: 70
kseniahrisanova@fedora:~/work/arch-pc/lab07$
```

рис. 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. 10).

```
kseniahrisonova@fedora:~/work/arch-pc/lab07 — mcedit lab7-2.lst
lab7-2.lst [----] 0 L: [ 1+ 0 1/217] *(0 /12937b) 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> nextchar:-----
8 <1> cmp byte [eax], 0
9 <1> jz finished
10 <1> inc eax
11 <1> jmp nextchar
12 <1> finished:
13 <1> sub eax, ebx
14 <1> pop ebx
15 <1> ret
16 <1>
17 <1>
18 <1> ;----- sprintf -----
19 <1> ; Функция печати сообщения
20 <1> ; входные данные: mov eax, <message>
21 <1> sprintf:
22 <1> push edx
23 <1> push ecx
24 <1> push ebx
25 <1> push eax
26 <1> call slen
27 <1>
28 <1> mov edx, eax
29 <1> pop eax
30 <1>
31 <1> mov ecx, eax
32 <1> mov ebx, 1
33 <1> mov eax, 4
34 <1> int 80h
35 <1>
36 <1> pop ebx
37 <1> pop ecx
38 <1> pop edx
39 <1> ret
40 <1>
41 <1>
42 <1> ;----- sprintf -----
43 <1> ; Функция печати сообщения с переводом строки
44 <1>
```

Рис. 10: Проверка файла листинга

Файл листинга NASM представляет собой расширенный вариант исходного текста программы, в котором для каждой строки указаны номер строки, смещение машинного кода, байты сгенерированных инструкций и исходная команда. Листинг позволяет анализировать соответствие команд ассемблера и результата их трансляции.

Удаляю один операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. 11).

```
GNU nano 2.9.3 /home/kseniahrisonova/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:
call sprint
mov edx, 10
call sread
mov eax, 8
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 iprintf
call quit
```

Рис. 11: Удаление операнда из программы

Создаётся только файл листинга lab7-2.lst, объектный файл не формируется из-за ошибки. В листинге у строки с испорченной инструкцией появляется сообщение об ошибке ассемблера (текст error: . . . после исходной строки).

(рис. 12).

```
kseniahrisonova@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:13: error: invalid combination of opcode and operands
kseniahrisonova@fedora:~/work/arch-pc/lab07$
```

Рис. 12: Просмотр ошибки в файле листинга

## 4.3 Задания для самостоятельной работы

Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 13).



The screenshot shows a terminal window with the nano text editor. The title bar indicates the user is kseniahrisanova@fedora, the current directory is /work/arch-pc/lab07, and the command being run is /usr/bin/mc -P /tmp/mc.pwd.9b9C89. The editor shows the file /home/kseniahrisanova/work/arch-pc/lab07/lab7-2.asm. The code is as follows:

```
GNU nano 8.3 /home/kseniahrisanova/work/arch-pc/lab07/lab7-2.asm
#include 'in_out.asm'

SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '17'
C dd '45'

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

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

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

mov ecx, [min]
спр ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx

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

The bottom of the window shows a menu bar with various icons and labels in Russian, such as 'Справка' (Help), 'Записать' (Save), 'Поиск' (Search), etc.

Рис. 13: программа самостоятельной работы

Текст первой программы:

```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '17'
C dd '45'
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]
```

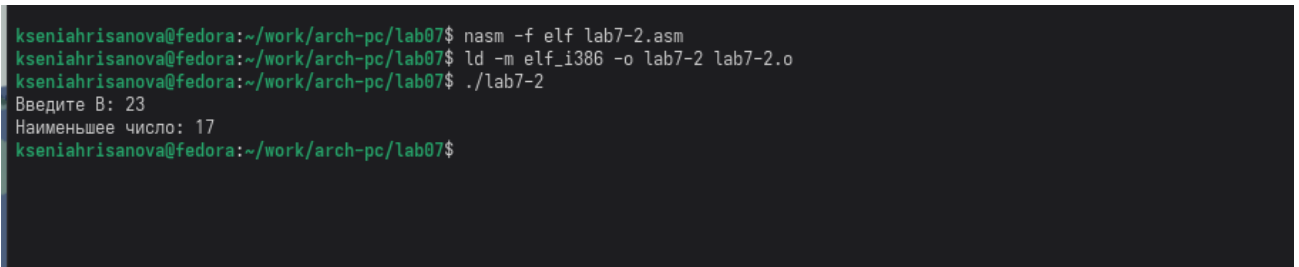


```

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

```

Проверяю работу первой программы (рис. 14).



```

kseniahrisnova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
kseniahrisnova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
kseniahrisnova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 23
Наименьшее число: 17
kseniahrisnova@fedora:~/work/arch-pc/lab07$

```

Рис. 14: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатуры переменных а и х (рис. 15).



```
GNU nano 8.3 /home/kseniahrisnova/work/arch-pc/lab07/lab7-3.asm
#include 'in_out.asm'

SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
msg_f: DB 'Результат f(x): ', 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
    jl less_than_a

    mov eax, 8
    jmp print_answer

less_than_a:
    mov eax, esi
    shl eax, 1
    sub eax, edi

print_answer:
    mov ebx, eax

    mov eax, msg_f
    call sprint
```

Рис. 15: Вторая программа самостоятельной работы

Текст второй программы:

```
%include 'in_out.asm'
SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
msg_f: DB 'Результат f(x): ', 0
```

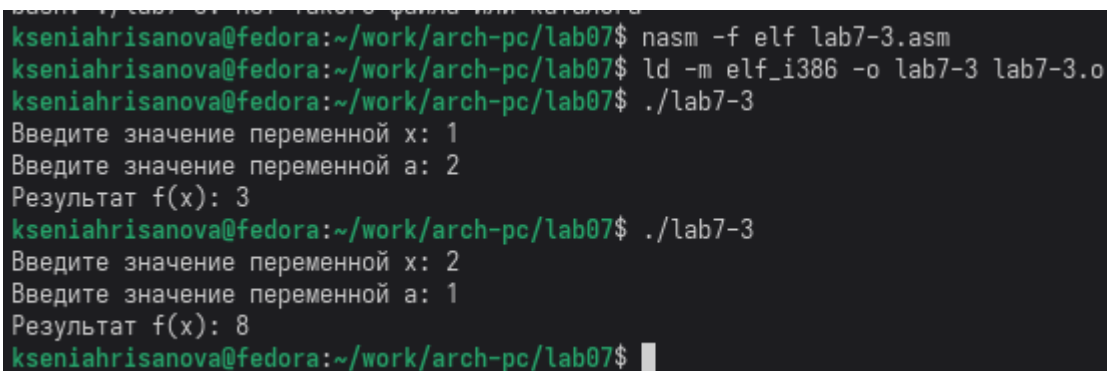
```
SECTION .bss
x: RESB 80
a: RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax, msg_x
    call sprint
    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi
    mov edi, eax
```

```

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
jl less_than_a
mov eax, 8
jmp print_answer
less_than_a:
mov eax, esi
shl eax, 1
sub eax, edi
print_answer:
mov ebx, eax
mov eax, msg_f
call sprint

```

Транслирую и компоную файл, запускаю и проверяю работу программы для значений а и х, которые даны в задании (рис. 16).



```

kseniahrisano@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
kseniahrisano@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
kseniahrisano@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 1
Введите значение переменной a: 2
Результат f(x): 3
kseniahrisano@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 2
Введите значение переменной a: 1
Результат f(x): 8
kseniahrisano@fedora:~/work/arch-pc/lab07$

```

Рис. 16: Проверка работы второй программы

## 5 Выводы

При выполнении лабораторной работы я изучила команды условных и безусловных переходов, приобрела навыки написания программ с использованием переходов и познакомилась с назначением и структурой файлов листинга.

## Список литературы

1. Курс на ТУИС
2. Лабораторная работа №7