

# **Отчет по лабораторной работе №7**

**Команды безусловного и условного переходов в Nasm.  
Программирование ветвлений.**

Жукова София Викторовна

# Содержание

<b>Цель работы</b>	<b>5</b>
<b>Задание</b>	<b>6</b>
<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>Выводы</b>	<b>18</b>

# Список иллюстраций

1	Создаем каталог и файл . . . . .	7
2	Заполняем файл . . . . .	8
3	Запускаем файл и смотрим на его работу . . . . .	8
4	Изменяем файл . . . . .	9
5	Проверяем работу файл . . . . .	9
6	Изменяем файл . . . . .	10
7	Запускаем файл . . . . .	10
8	Создаем файл . . . . .	10
9	Создаем файл . . . . .	11
10	Проверяем работу . . . . .	11
11	Проверем работу . . . . .	12
12	Создаем файл . . . . .	12
13	Открываем файл . . . . .	12
14	Открываем файл . . . . .	13
15	Строчки . . . . .	13
16	Открываем файл . . . . .	14
17	Трвансируем файл . . . . .	14
18	Проверяем . . . . .	14
19	Проверяем . . . . .	15
20	Создаем . . . . .	15
21	Заполняем файл . . . . .	16
22	Проверяем работу прораммы . . . . .	16
23	Создаем исполняемый файл . . . . .	17
24	Заполняем файл . . . . .	17
25	Проверяем работу файла . . . . .	17

## **Список таблиц**

## Цель работы

Изучить команды условного и безусловного переходов. Приобрести навыки написания программ с использованием переходов. Познакомиться с назначением и структурой файла листинга.

# Задание

Написать программы для решения системы выражений.

# Выполнение лабораторной работы

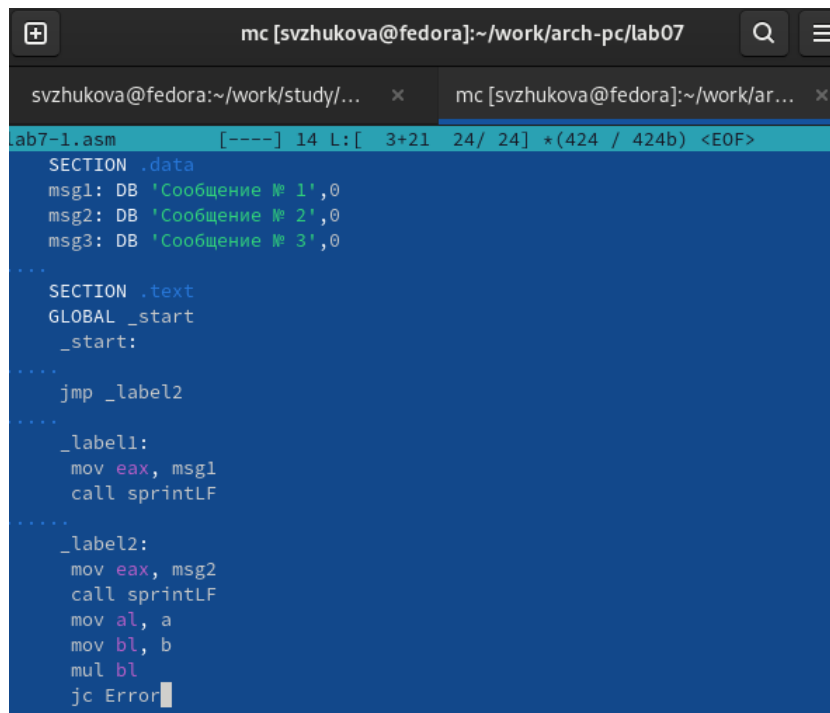
## Порядок выполнения лабораторной работы

**1. Реализация переходов в NASM** Создадим каталог для программ лабораторной работы № 7, перейдем в него и создадим файл lab7-1.asm (рис. [-@fig:001]).

```
4/report$ mkdir ~/work/arch-pc/lab07
svzhukova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07$
4/report$ cd ~/work/arch-pc/lab07
svzhukova@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
svzhukova@fedora:~/work/arch-pc/lab07$
```

Рис. 1: Создаем каталог и файл

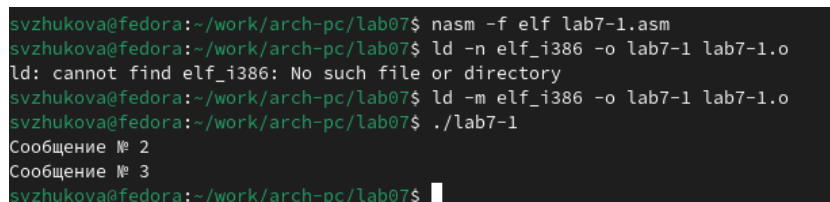
Откроем файл lab7-1.asm в Midnight Commander и введем текст программы из листинга 7.1. (рис. [-@fig:002]).



```
lab7-1.asm [----] 14 L: [ 3+21 24/ 24] *(424 / 424b) <EOF>
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
...
_label2:
mov eax, msg2
call sprintf
mov al, a
mov bl, b
mul bl
jc Error
```

Рис. 2: Заполняем файл

Создадим исполняемый файл и запустим его. (рис. [-@fig:003]).



```
svzhukova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
svzhukova@fedora:~/work/arch-pc/lab07$ ld -n elf_i386 -o lab7-1 lab7-1.o
ld: cannot find elf_i386: No such file or directory
svzhukova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
svzhukova@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
svzhukova@fedora:~/work/arch-pc/lab07$
```

Рис. 3: Запускаем файл и смотрим на его работу

Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. (рис. [-@fig:004]).



```

lab7-1.asm  [----]  6 L:[ 3+26 29/ 29] *(483 / 492b) 0099 0
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: Изменяем файл

Создадим исполняемый файл и проверим его работу. (рис. [-@fig:005]).

```

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

```

Рис. 5: Проверяем работу файл

Снова открываем файл для редактирования и изменяем его (рис. [-@fig:006]).

```

lab7-1.asm      [----] 15 L: [ 4+26 30/ 30] *(510 / 510b) <EOI
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. 6: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. [-@fig:007]).

```

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

```

Рис. 7: Запускаем файл

Создадим файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. (рис. [-@fig:008]).

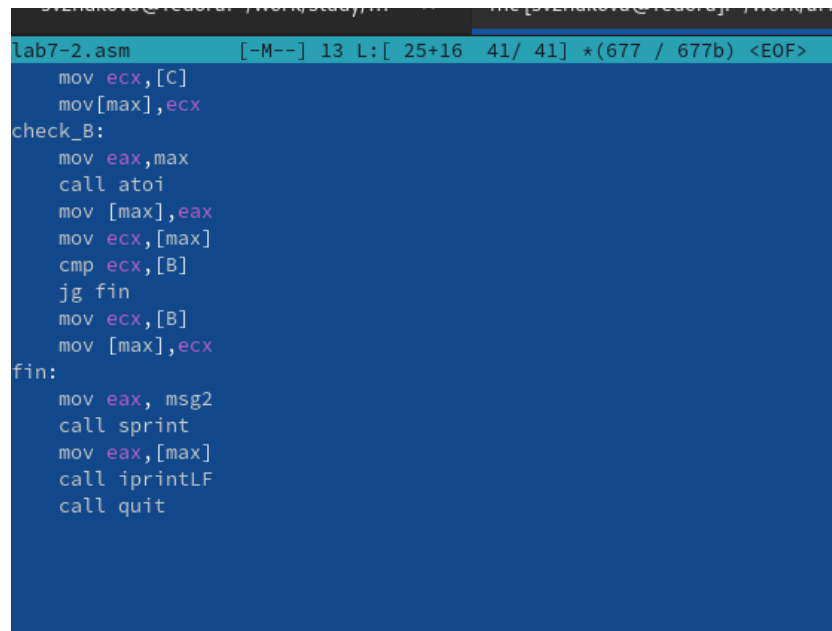
```

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

```

Рис. 8: Создаем файл

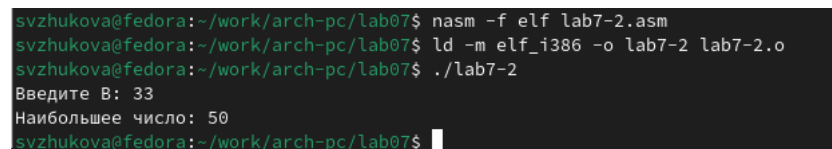
Внимательно изучим текст программы из листинга 7.3 и введем в lab7-2.asm.  
(рис. [-@fig:009]).



```
lab7-2.asm [-M--] 13 L: [ 25+16 41/ 41] *(677 / 677b) <EOF>
    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. 9: Создаем файл

Создадим исполняемый файл и проверим его работу для разных значений В.  
(рис. [-@fig:010]).



```
svzhukova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
svzhukova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
svzhukova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 33
Наибольшее число: 50
svzhukova@fedora:~/work/arch-pc/lab07$
```

Рис. 10: Проверяем работу

(рис. [-@fig:012]).

```
svzhukova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 22
Наибольшее число: 50
svzhukova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 1
Наибольшее число: 50
svzhukova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 55
Наибольшее число: 55
```

Рис. 11: Проверем работу

## 2. Изучение структуры файлы листинга

Создадим файл листинга для программы из файла lab7-2.asm (рис. [-@fig:011]).

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

Рис. 12: Создаем файл

Открываем файл листинга с помощью команды mcedit (рис. [-@fig:013]).

```
svzhukova@fedora:~/work/arch-pc/lab07$ mcedit lab7-2.lst
```

Рис. 13: Открываем файл

Изучаем файл (рис. [-@fig:014]).

```

1      %include 'in_out.asm'
2      <1> ;----- slen -----
3      <1> ; Функция вычисления длины сообщения
4      <1> slen:.....
5      00000000 53      <1>      push     ebx.....
6      00000001 89C3    <1>      mov      ebx, eax.....
7      <1>.....
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

```

Рис. 14: Открываем файл

(рис. [-@fig:015]).

```

51 00000038 50      <1>      push     eax
52 00000039 89E0    <1>      mov      eax, esp
53 0000003B E8CFFFFFFF <1>      call     sprint

```

Рис. 15: Строчки

51: 00000038-адрес в сегменте кода, 50-машинный код, mov eax, 0AH- копируем значение переменой 0AH в eax 52: 00000039-адрес в сегменте кода, 89E0-машинный код, push eax присвоение переменной eax значения 53: 0000003B-адрес в сегменте кода, E8CFFFFFFF-машинный код, call sprint вызывает функцию

Откроем файл с программой lab7-2.asm и в инструкции с двумя операндами удалим один операнд. (рис. [-@fig:016]).

```

%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,
    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

```

Рис. 16: Открываем файл

Выполним трансляцию с получением файла листинга: (рис. [-@fig:017]).

```

svzhukova@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:16: error: invalid combination of opcode and operands
svzhukova@fedora:~/work/arch-pc/lab07$

```

Рис. 17: Транслируем файл

При трансляции файла, выдается ошибка, но создаются исполняемый файл lab7-2 и lab7-2.lst (рис. [-@fig:018]).

```

svzhukova@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2  lab7-2.asm  lab7-2.lst

```

Рис. 18: Проверяем

Просматриваем файл листинга (рис. [-@fig:019]).

```

svzhukova@fedora:~/work/study/... x svzhukova@fedora:~/work/arch-p... x
lab7-2.lst [----] 40 L:[ 31+13 44/218] *(2640/13160b) 0059 0x03B [*][X]
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
45 <1> ; входные данные: mov eax,<message>
46 <1> sprintf:
47 0000002D E8DDFFFFFF <1> call sprintf
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

```

Рис. 19: Проверяем

### Задание для самостоятельной работы

ВАРИАНТ 7 1. Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Создадим исполняемый файл (рис. [-@fig:020]).

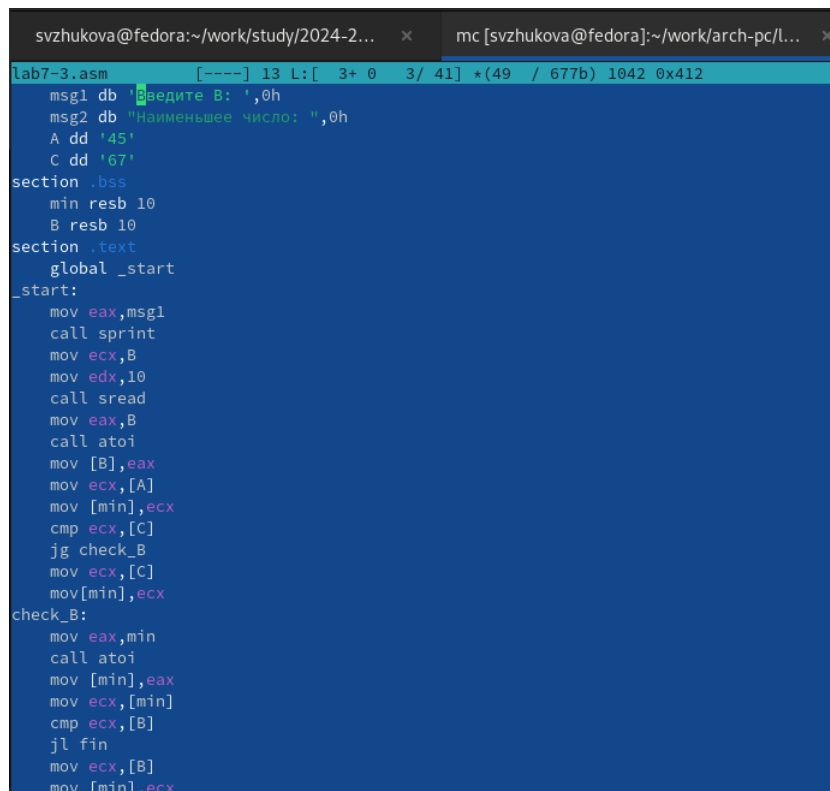
```

svzhukova@fedora:~/work/arch-pc/lab07$ touch lab7-3.asm
svzhukova@fedora:~/work/arch-pc/lab07$

```

Рис. 20: Создаем

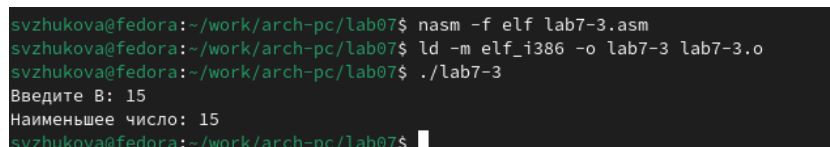
Открываем файл и пишем программу, которая выберет наименьшую из трех переменных. (рис. [-@fig:021]).



```
lab7-3.asm [----] 13 L: [ 3+ 0 3/ 41] *(49 / 677b) 1042 0x412
msg1 db 'Введите B: ',0h
msg2 db "Наименьшее число: ",0h
A dd '45'
C dd '67'
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]
jl fin
mov ecx,[B]
mov [min],ecx
```

Рис. 21: Заполняем файл

Транслируем файл и проверяем программу (рис. [-@fig:022]).



```
svzhukova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
svzhukova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
svzhukova@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 15
Наименьшее число: 15
svzhukova@fedora:~/work/arch-pc/lab07$
```

Рис. 22: Проверяем работу прораммы

2. Напишите программу, которая для введенных с клавиатуры значений  $x$  и  $a$  вычисляет значение заданной функции  $f(x)$  и выводит результат вычислений.

Создаем исполняемый файл (рис. [-@fig:023]).



```
svzhukova@fedora:~/work/arch-pc/lab07$ touch lab7-4.asm
svzhukova@fedora:~/work/arch-pc/lab07$
```

Рис. 23: Создаем исполняемый файл

Заполняем файл (рис. [-@fig:024]).

```
svzhukova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab...
lab7-4.asm [----] 13 L: [ 1+ 9 10/ 44] *(206 / 712b) 0010 0x00A
#include 'in_out.asm'
section .data
    msg1 db 'Введите x: ',0h
    msg2 db 'Введите a: ',0h
    otv: DB 'F(x) = ',0h
section .bss
    x: RESB 80
    a: RESB 80
    res: RESB 80
section .text
    global _start
_start:
    mov eax,msg1
    call sprint
    mov ecx,x
    mov edx,80
    call sread
    mov eax,x
    call atoi
    mov [x],eax
    mov eax,msg2
    call sprint
    mov ecx,a
    mov edx,80
    call sread
    mov eax,a
    call atoi
    mov [a],eax
    cmp ecx,[x]
    jg check_A
    mov ecx,[x]
    sub ecx,[a]
    mov [res],ecx
```

Рис. 24: Заполняем файл

Транслируем и проверяем работу файла для разных значений (рис. [-@fig:025]).

```
svzhukova@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 1
Введите a: 1
F(x) = 6
svzhukova@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 2
Введите a: 1
F(x) = 6
svzhukova@fedora:~/work/arch-pc/lab07$
```

Рис. 25: Проверяем работу файла

## Выводы

Мы изучили команды условного и безусловного переходов, приобрели навыки написания программ с использованием переходов, познакомились с назначением и структурой файла листинга.