

# **Команды безусловного и условного переходов в NASM. Программирование ветвлений**

**Лабораторная работа №7**

Владимир Романович Козомазов

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Описание выполняемого задания . . . . .	6
2.2	Выполнение заданий лабораторной работы . . . . .	6
2.2.1	Реализация переходов в NASM . . . . .	6
2.2.2	Изучение структуры файла листинга . . . . .	11
2.2.3	Трансляция кода программы и изучение листинга с ошибкой .	12
2.3	Выводы по результатам выполнения заданий лабораторной работы	14
<b>3</b>	<b>Выполнение самостоятельного задания</b>	<b>15</b>
3.1	Описание выполняемого самостоятельного задания . . . . .	15
3.2	Выполнение заданий для самостоятельной работы . . . . .	15
3.2.1	Выполнение задания 1 . . . . .	15
3.2.2	Выполнение задания 2 . . . . .	16
3.3	Выводы по результатам выполнения самостоятельного задания . .	17
<b>4</b>	<b>Выводы</b>	<b>19</b>
<b>5</b>	<b>Листинги написанных программ</b>	<b>20</b>
5.1	Программа lab7-1.asm . . . . .	20
5.2	Программа lab7-2.asm . . . . .	21
5.3	Программа lab7_min_of_three.asm . . . . .	23
5.4	Программа lab7_calc_function.asm . . . . .	26

# Список иллюстраций

2.1	создание каталога для лабораторной работы и создание файла lab7-1.asm . . . . .	6
2.2	Ввод кода в файл lab7-1.asm из листинга . . . . .	7
2.3	Компиляция и запуск программы lab7-1 . . . . .	7
2.4	Изменение кода программы lab7-1 . . . . .	8
2.5	Компиляция и запуск программы lab7-1 . . . . .	8
2.6	Изменение кода программы lab7-1 в соответствии с заданием . . . .	9
2.7	Запуск программы lab7-1 и получение результатов, соответствующих заданию . . . . .	9
2.8	Создание файла lab7-2.asm и запись в него кода . . . . .	10
2.9	Компиляция и запуск программы lab7-2, проверка работы программы для разных значений В . . . . .	10
2.10	Создание файла листинга lab7-2.lst и его открытие . . . . .	11
2.11	Объяснение строки 1 в файле листинга lab7-2.lst . . . . .	11
2.12	Объяснение строки 194 в файле листинга lab7-2.lst . . . . .	12
2.13	Объяснение строки 206 в файле листинга lab7-2.lst . . . . .	12
2.14	Удаление операнда в инструкции cmp в файле lab7-2.asm . . . . .	13
2.15	Создание листинга для файла lab7-2.asm с ошибкой . . . . .	13
2.16	Открытие листинга файла lab7-2.asm, созданный с ошибкой . . . .	14
3.1	Написание программы lab7_min_of_three.asm . . . . .	16
3.2	Проверка работы программы lab7_min_of_three.asm . . . . .	16
3.3	Написание программы lab7_calc_function.asm . . . . .	17
3.4	Проверка работы программы lab7_calc_function.asm . . . . .	17

## **Список таблиц**

# 1 Цель работы

- Изучение команд условного и безусловного переходов в языке ассемблера NASM и приобретение навыков написания программ с использованием ветвлений.
- Знакомство с назначением и структурой файла листинга.

## 2 Выполнение лабораторной работы

### 2.1 Описание выполняемого задания

- Написать программу, реализующую безусловный переход `jmp` в NASM и изменить ее в соответствии с заданием.
- Написать программу, реализующую условный переход `jg` в NASM и вычисляющую максимум из трех чисел, два из которых заданы, а третье вводится с клавиатуры. Получить файл листинга и исследовать его структуру.

### 2.2 Выполнение заданий лабораторной работы

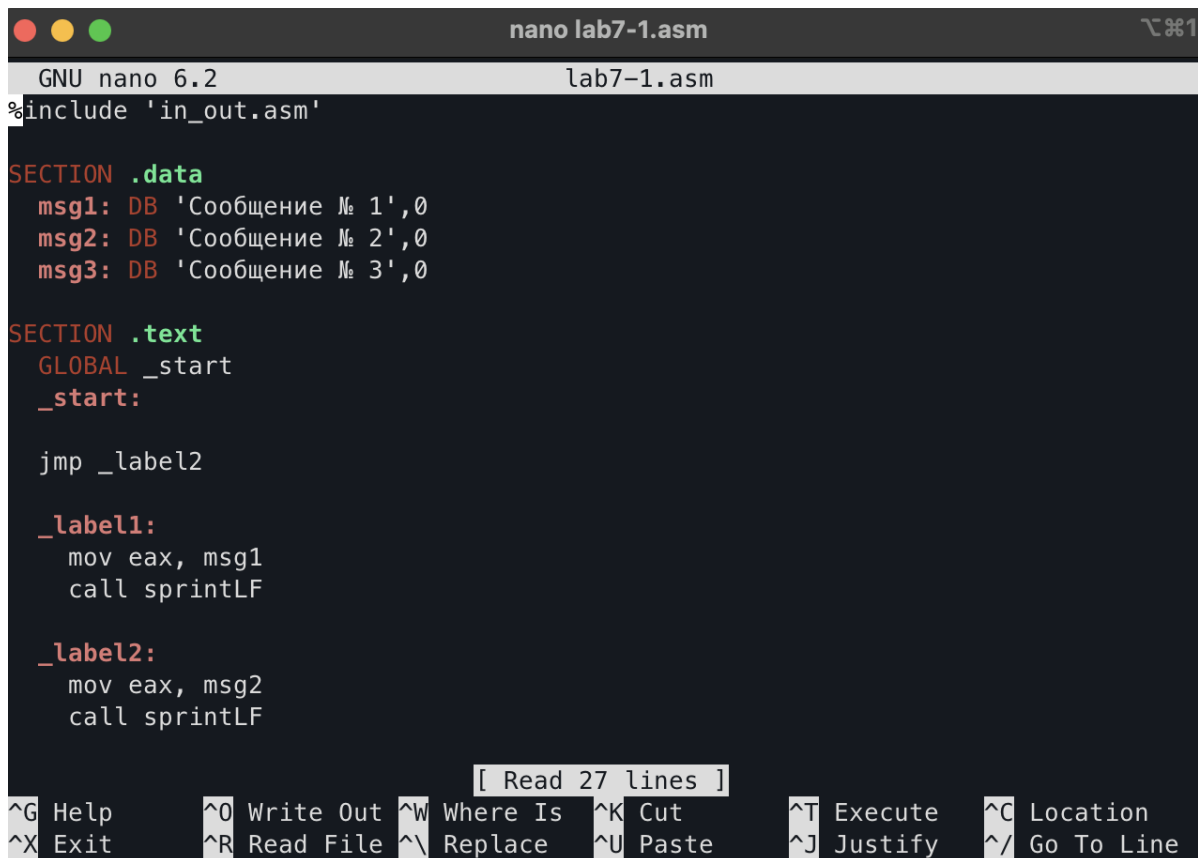
#### 2.2.1 Реализация переходов в NASM

Создал каталог для программы лабораторной работы №7 и перешёл в него.  
(рис. 2.1)

```
> mkdir ~/work/arch-pc/lab07  
> cd ~/work/arch-pc/lab07  
> touch lab7-1.asm
```

Рис. 2.1: создание каталога для лабораторной работы и создание файла `lab7-1.asm`

Ввёл код в файл `lab7-1.asm` из листинга 7.1 (рис. 2.2)



```
nano lab7-1.asm
GNU nano 6.2 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

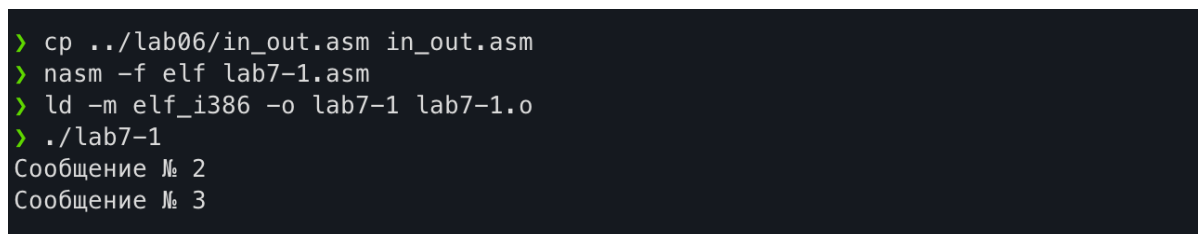
_label1:
mov eax, msg1
call sprintLF

_label2:
mov eax, msg2
call sprintLF

[ Read 27 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

Рис. 2.2: Ввод кода в файл lab7-1.asm из листинга

Скомпилировал и запустил программу lab7-1 (рис. 2.3)



```
> cp ../lab06/in_out.asm in_out.asm
> nasm -f elf lab7-1.asm
> ld -m elf_i386 -o lab7-1 lab7-1.o
> ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 2.3: Компиляция и запуск программы lab7-1

Изменил код программы lab7-1 из листинга 7.2 (рис. 2.4)

```

GNU nano 6.2                                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

[ Read 29 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line

```

Рис. 2.4: Изменение кода программы lab7-1

Скомпилировал и запустил программу lab7-1 с кодом из листинга 7.2 (рис. 2.5)

```

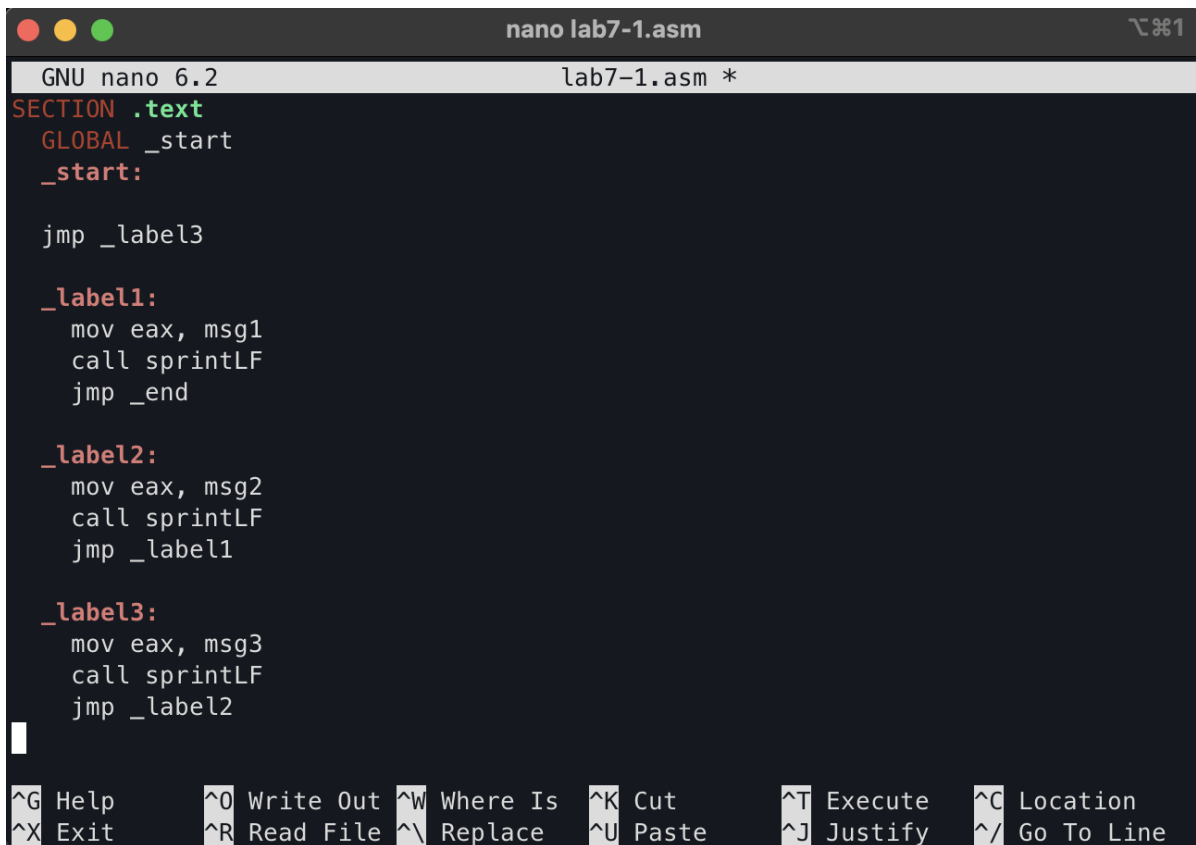
> nasm -f elf lab7-1.asm
> ld -m elf_i386 -o lab7-1 lab7-1.o
> ./lab7-1
Сообщение № 2
Сообщение № 1

```

Рис. 2.5: Компиляция и запуск программы lab7-1

Изменил код программы lab7-1, чтобы вывод программы соответствовал заданию (рис. 2.6)





```
GNU nano 6.2                                lab7-1.asm *
SECTION .text
GLOBAL _start
_start:

jmp _label3

_label1:
mov eax, msg1
call sprintf
jmp _end

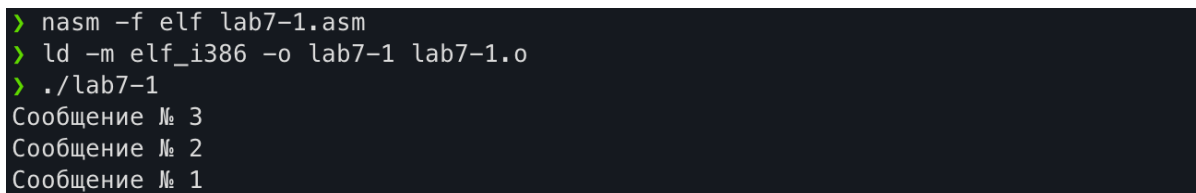
_label2:
mov eax, msg2
call sprintf
jmp _label1

_label3:
mov eax, msg3
call sprintf
jmp _label2
```

^G Help   ^O Write Out   ^W Where Is   ^K Cut   ^T Execute   ^C Location  
^X Exit   ^R Read File   ^\ Replace   ^U Paste   ^J Justify   ^/ Go To Line

Рис. 2.6: Изменение кода программы lab7-1 в соответствии с заданием

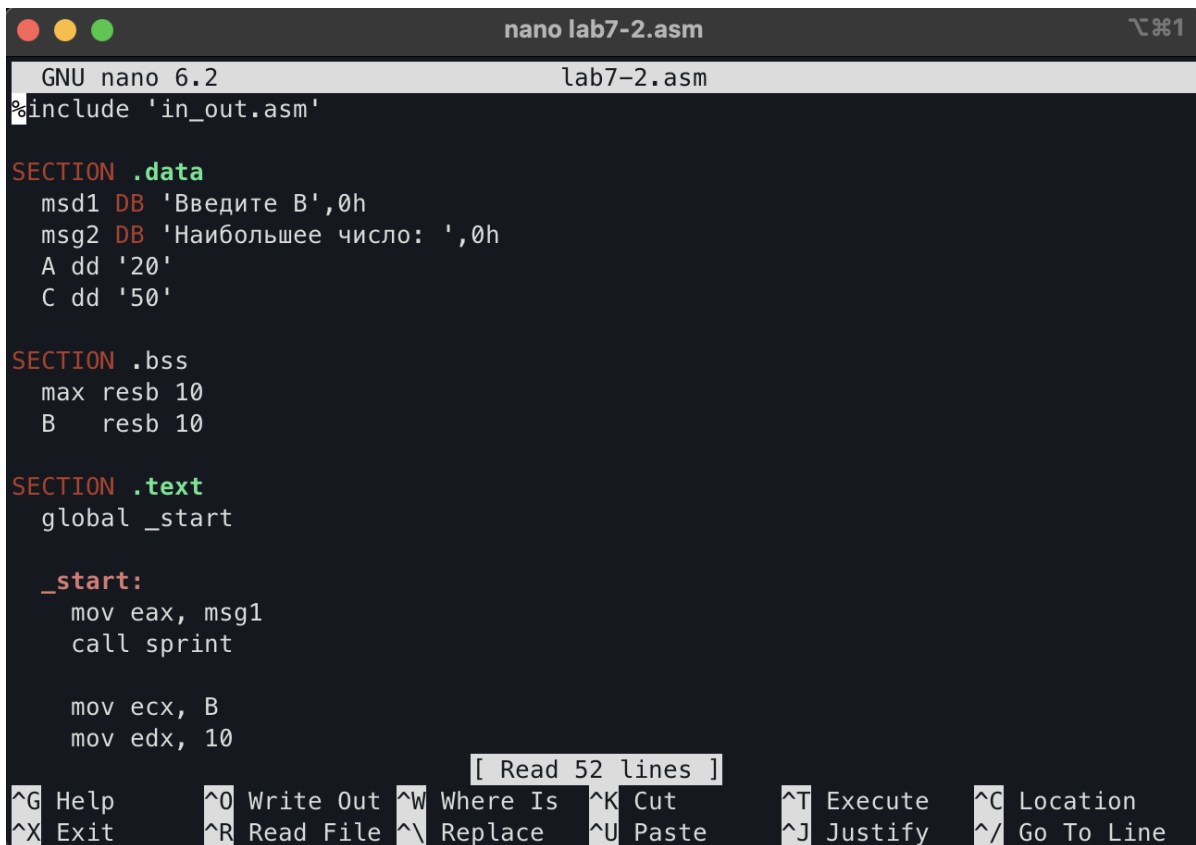
Снова скомпилировал и запустил программу lab7-1 и получил результаты, соответствующие заданию (рис. 2.7)



```
> nasm -f elf lab7-1.asm
> ld -m elf_i386 -o lab7-1 lab7-1.o
> ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 2.7: Запуск программы lab7-1 и получение результатов, соответствующих заданию

Создал файл lab7-2.asm и записал код из листинга 7.3 (рис. 2.8)



```
nano lab7-2.asm
GNU nano 6.2 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

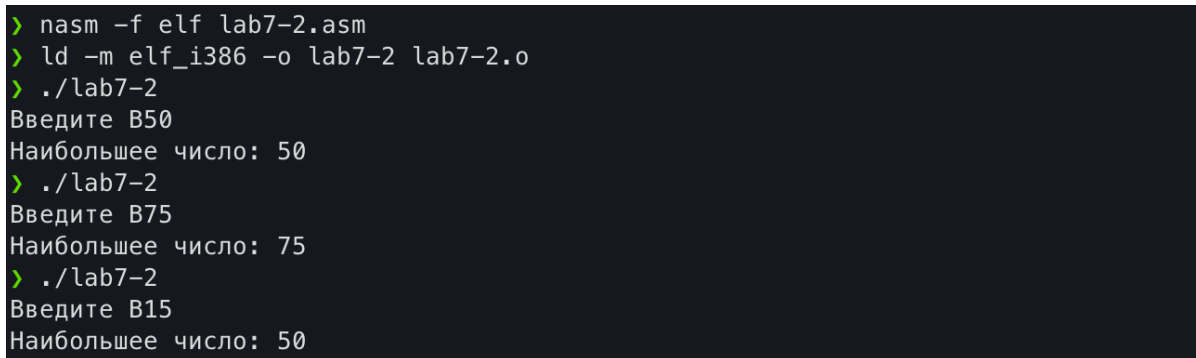
_start:
    mov eax, msg1
    call sprint

    mov ecx, B
    mov edx, 10

[ Read 52 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

Рис. 2.8: Создание файла lab7-2.asm и запись в него кода

Скомпилировал и запустил программу lab7-2 с кодом из листинга 7.3 (рис. 2.9)



```
> nasm -f elf lab7-2.asm
> ld -m elf_i386 -o lab7-2 lab7-2.o
> ./lab7-2
Введите B50
Наибольшее число: 50
> ./lab7-2
Введите B75
Наибольшее число: 75
> ./lab7-2
Введите B15
Наибольшее число: 50
```

Рис. 2.9: Компиляция и запуск программы lab7-2, проверка работы программы для разных значений B

## 2.2.2 Изучение структуры файла листинга

Создал файл листинга lab7-2.lst и открыл его при помощи текстового редактора mcedit (рис. 2.10)

```
> nasm -f elf -l lab7-2.lst lab7-2.asm
> mcedit lab7-2.lst
```

Рис. 2.10: Создание файла листинга lab7-2.lst и его открытие

### Объяснение трех строк файла листинга

Строка листинга 1:

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

Эта строка является макрокомандой включения файла, она включает весь код из файла in\_out.asm в файл lab7-2.asm, что мы и видим в файле листинга (рис. 2.11)

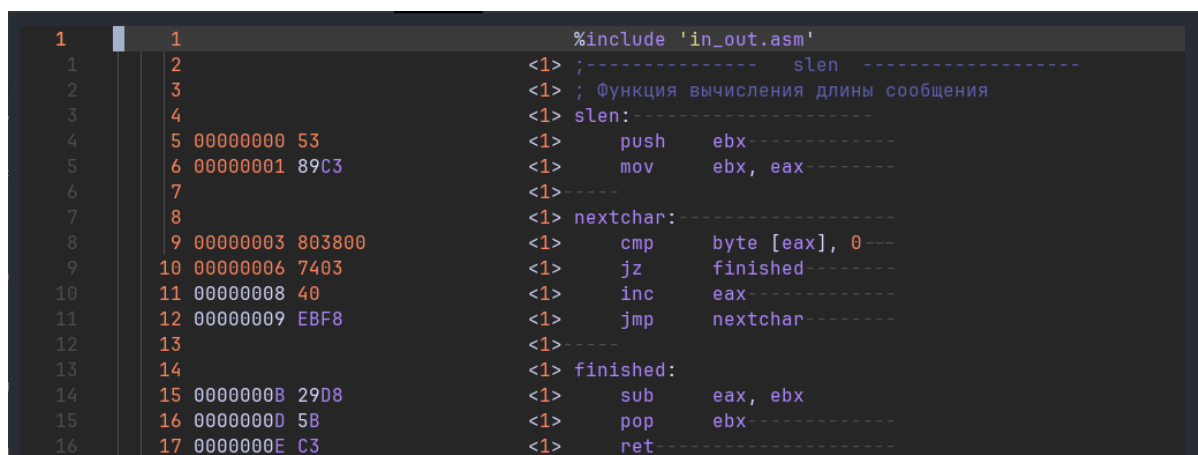


Рис. 2.11: Объяснение строки 1 в файле листинга lab7-2.lst

Строка листинга 194:

```
mov ecx, B
```

Данная строка соответствует строке 20 исходного файла (номер строки находится в столбце 1). Далее в столбце 2 находится адрес смещения (offset). В столбце

3 находится машинный код выполняемой инструкции и данные, а в столбце 4 находится собственно сам ассемблерный код (рис. 2.12)

```

11      9                                SECTION .bss
10      10 00000000 <res Ah>             max resb 10
9       11 0000000A <res Ah>             B   resb 10
8       12 -----
7       13                                SECTION .text
6       14                                global _start
5       15 -----
4       16                                _start:
3       17 000000E8 B8[00000000]          mov eax, msg1
2       18 000000ED E81DFFFFFF           call sprint
1       19 -----
194     20 000000F2 B9[0A000000]          mov ecx, B
1       21 000000F7 BA0A000000           mov edx, 10
2       22 000000FC E842FFFFFF           call sread
3       23 -----
4       24 00000101 B8[0A000000]          mov eax, B
5       25 00000106 E891FFFFFF           call atoi

```

Рис. 2.12: Объяснение строки 194 в файле листинга lab7-2.lst

Строка листинга 206:

**jg** check\_B

В первом столбце этой строки находится номер строки исходного файла, во втором столбце находится адрес смещения, в третьем столбце находится машинный код выполняемой инструкции, а в четвертом столбце собственно сам ассемблерный код (рис. 2.13)

```

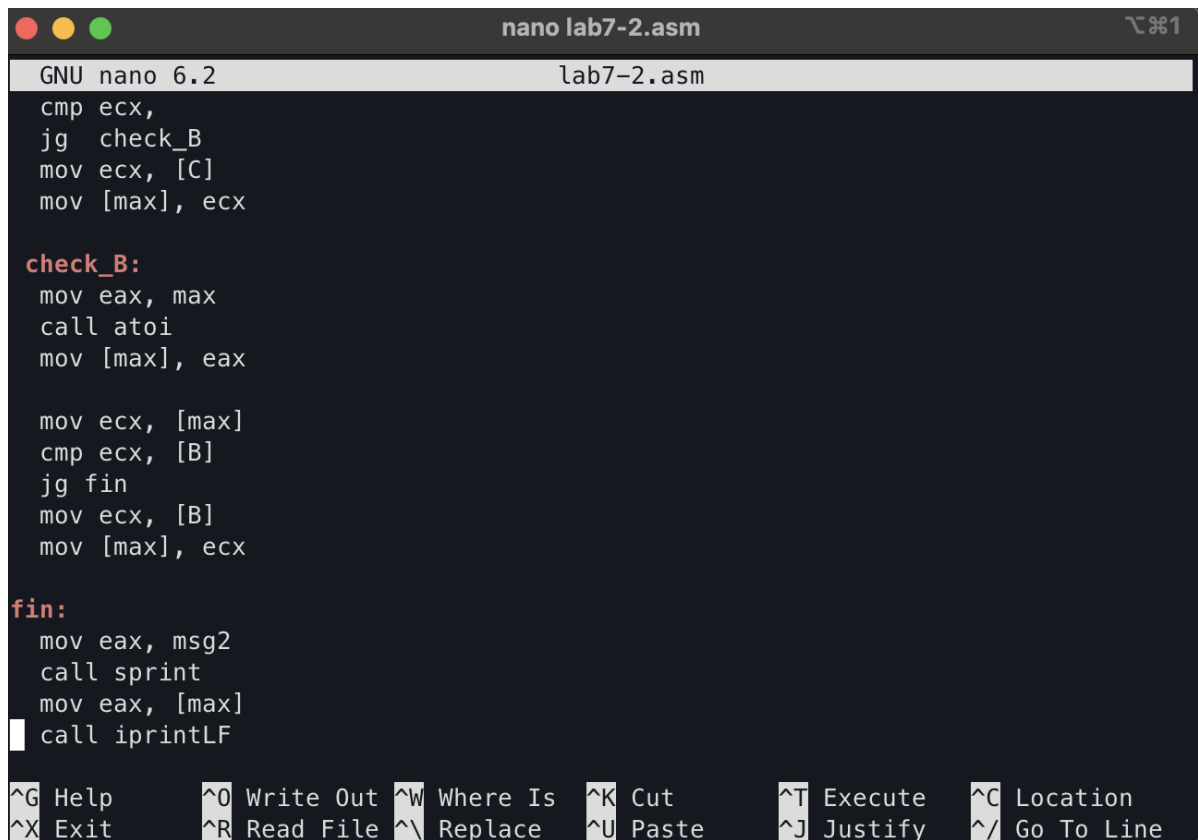
5       27 -----
4       28 00000110 8B0D[34000000]        mov ecx, [A]
3       29 00000116 890D[00000000]        mov [max], ecx
2       30 -----
1       31 0000011C 3B0D[38000000]        cmp ecx, [C]
206     32 00000122 7F0C                   jg check_B
1       33 00000124 8B0D[38000000]        mov ecx,[C]
2       34 0000012A 890D[00000000]        mov [max],ecx
3       35 -----
4       36                                check_B:
5       37 00000130 B8[00000000]          mov eax, max

```

Рис. 2.13: Объяснение строки 206 в файле листинга lab7-2.lst

### 2.2.3 Трансляция кода программы и изучение листинга с ошибкой

Изменил инструкцию `cmp`, удалив один операнд в файле `lab7-2.asm` (рис. 2.14)



```
GNU nano 6.2 lab7-2.asm
cmp ecx,
jg check_B
mov ecx, [C]
mov [max], ecx

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

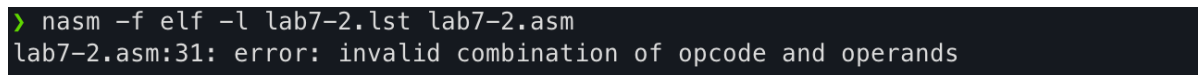
mov ecx, [max]
cmp ecx, [B]
jg fin
mov ecx, [B]
mov [max], ecx

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

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute  ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify  ^_ Go To Line
```

Рис. 2.14: Удаление операнда в инструкции cmp в файле lab7-2.asm

Выполнил трансляцию с получением файла листига (рис. 2.15)



```
> nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:31: error: invalid combination of opcode and operands
```

Рис. 2.15: Создание листинга для файла lab7-2.asm с ошибкой

Проверил листинг файла lab7-2.asm, созданный с ошибкой и заметил изменения данных в листинге (рис. 2.16). В листинге добавилась строка с данными об ошибке.

```

mcedit lab7-2.lst
/home/vo~7-2.lst  [----]  0 L: [206+22 228/230] *(13516/13609b) 0032 0x020[*] [X]
31                                     cmp ecx,
31          *****                  error: invalid combination of opcode an
32 0000011C 7F0C                      jg check_B
33 0000011E 8B0D[3A000000]             mov ecx, [C]
34 00000124 890D[00000000]             mov [max], ecx
35.....
36                                     check_B:
37 0000012A B8[00000000]               mov eax, max
38 0000012F E868FFFFFF                 call atoi
39 00000134 A3[00000000]               mov [max], eax
40.....
41 00000139 8B0D[00000000]             mov ecx, [max]
42 0000013F 3B0D[0A000000]             cmp ecx, [B]
43 00000145 7F0C                      jg fin
44 00000147 8B0D[0A000000]             mov ecx, [B]
45 0000014D 890D[00000000]             mov [max], ecx
46.....
47                                     fin:
48 00000153 B8[14000000]               mov eax, msg2
49 00000158 E8B2FEFFFF                 call sprint
50 0000015D A1[00000000]               mov eax, [max]
51 00000162 E81FFFFFFF                 call iprintLF
52 00000167 E86FFFFFFF                 call quit
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit

```

Рис. 2.16: Открытие листинга файла lab7-2.asm, созданный с ошибкой

Вывод: когда в ассемблерном коде допущена ошибка, то при трансляции объектный файл не создается, а выводится сообщение об ошибке. Файл листинга создается и в него выводится строка с данными об ошибке.

## 2.3 Выводы по результатам выполнения заданий лабораторной работы

При выполнении заданий лабораторной работы были написаны программы, с помощью которых были изучены инструкции безусловного и условного переходов. Также были получен файл листинга и изучена его структура.

## 3 Выполнение самостоятельного задания

### 3.1 Описание выполняемого самостоятельного задания

Задание для самостоятельной работы состоит из двух частей: \* Написать программу нахождения наименьшего из трех целых чисел. \* Написать программу вычисления функции

$$f(x) = \begin{cases} a - 7, & a \geq 7 \\ ax, & a < 7 \end{cases}$$

в зависимости от вводимого с клавиатуры значения  $x$  и параметра  $a$ .

### 3.2 Выполнение заданий для самостоятельной работы

#### 3.2.1 Выполнение задания 1

Для программы нахождения наименьшего из трех чисел создал файл `lab7_min_of_three.asm` и ввел в него код (рис. 3.1)

```

1 include "in_out.asm"
2
3 section .data
4     msg_input_a db "Введите a: ", 0h
5     msg_input_b db "Введите b: ", 0h
6     msg_input_c db "Введите c: ", 0h
7     msg_result  db "Наименьшее число: ", 0h
8
9 section .bss
10    a          resb 10
11    b          resb 10
12    c          resb 10
13    min        resb 10
14
15 section .text
16     global _start
17
18     _start:
19         ; Запрашиваем значение переменной a
20         mov     eax, msg_input_a

```

NORMAL ASM min\_of\_three.asm ~@k < 25 Top 1:1 07:13

Рис. 3.1: Написание программы lab7\_min\_of\_three.asm

Проверил работу программы lab7\_min\_of\_three на данных из задания (рис. 3.2)

```

> nasm -f elf lab7_min_of_three.asm
> ld -m elf_i386 -o lab7_min_of_three lab7_min_of_three.o
> ./lab7_min_of_three
Введите a: 84
Введите b: 32
Введите c: 77
Наименьшее число: 32

```

Рис. 3.2: Проверка работы программы lab7\_min\_of\_three.asm

### 3.2.2 Выполнение задания 2

Для программы вычисления функции  $f(x)$  создал файл lab7\_calc\_function.asm и ввел в него соответствующий код (рис. 3.3)



```

1  %include "in_out.asm"
2
3  section .data
4      msg_input_x    db    "Введите x: ", 0h
5      msg_input_a    db    "Введите a: ", 0h
6      msg_result     db    "Результат: ", 0h
7
8  section .bss
9      x              resb  10
10     a              resb  10
11     result         resb  10
12
13  section .text
14      global _start
15
16  _start:
17      ; Запрашиваем значение переменной x
18      mov     eax, msg_input_x
19      call    sprint

```

Рис. 3.3: Написание программы lab7\_calc\_function.asm

Проверка работы программы lab7\_calc\_function на данных из задания (рис. 3.4)

```

> nasm -f elf lab7_calc_function.asm
> ld -m elf_i386 -o lab7_calc_function lab7_calc_function.o
> ./lab7_calc_function
Введите x: 3
Введите a: 9
Результат: 2

```

Рис. 3.4: Проверка работы программы lab7\_calc\_function.asm

### 3.3 Выводы по результатам выполнения

#### самостоятельного задания

В ходе выполнения самостоятельного задания были написаны программы, которые вычисляют наименьшее из трех чисел и вычисляют функцию  $f(x)$  в зависимости от вводимого с клавиатуры значения  $x$  и параметра  $a$ . Были исполь-

зованы методы сравнения и ветвлений, освоенные в ходе выполнения лабораторной работы.

## 4 Выводы

- В ходе выполнения лабораторной работы были изучены команды безусловного и условного переходов в языке ассемблера NASM, приобретены навыки написания программ с использованием ветвлений, а также написаны несколько программ, использующих ветвления.
- При трансляции программы был получен файл листинга, изучено содержание файла листинга и его структура.

## 5 Листинги написанных программ

### 5.1 Программа 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 sprintLF
    jmp _label1

_label3:
    mov eax, msg3
    call sprintLF
    jmp _label2

_end:
    call quit

```

## 5.2 Программа 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,
```

```
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

```

### 5.3 Программа lab7\_min\_of\_three.asm

```

#include "in_out.asm"

section .data
    msg_input_a    db    "Введите a: ", 0h
    msg_input_b    db    "Введите b: ", 0h
    msg_input_c    db    "Введите c: ", 0h
    msg_result     db    "Наименьшее число: ", 0h

section .bss
    a              resb 10
    b              resb 10
    c              resb 10
    min            resb 10

section .text
    global _start

_start:
    ; Запрашиваем значение переменной a

```

```

mov    eax, msg_input_a
call  sprint
; Организует ввод с клавиатуры значения переменной a
mov    ecx, a
mov    edx, 10
call  sread
; Преобразует строку, полученную с клавиатуры, в число
mov    eax, a
call  atoi
mov    [a], eax

; Запрашиваем значение переменной b
mov    eax, msg_input_b
call  sprint
; Организует ввод с клавиатуры значения переменной b
mov    ecx, b
mov    edx, 10
call  sread
; Преобразует строку, полученную с клавиатуры, в число
mov    eax, b
call  atoi
mov    [b], eax

; Запрашиваем значение переменной c
mov    eax, msg_input_c
call  sprint
; Организует ввод с клавиатуры значения переменной a
mov    ecx, c
mov    edx, 10

```



```

call    sread
; Преобразуем строку, полученную с клавиатуры, в число
mov     eax, c
call    atoi
mov     [c], eax

; min = a
mov     ecx, [a]
mov     [min], ecx

; Сравниваем a и b
cmp     ecx, [b]
; Если a > b, то min = a и переходим
; к сравнению min и c
jl      check_c
; Иначе min = b
mov     ecx, [b]
mov     [min], ecx

; Сравниваем min(a,b) и c
check_c:
mov     ecx, [min]
cmp     ecx, [c]
; Если min < c, то уже нашли min
jl      fin
; Иначе min = c
mov     ecx, [c]
mov     [min], ecx

```

```

fin:
    ; Выводим сообщение пользователю
    mov     eax, msg_result
    call    sprint
    ; Выводим результат
    mov     eax, [min]
    call    iprintLF

    call    quit

```

## 5.4 Программа lab7\_calc\_function.asm

```

%include "in_out.asm"

section .data
    msg_input_x    db    "Введите x: ", 0h
    msg_input_a    db    "Введите a: ", 0h
    msg_result     db    "Результат: ", 0h

section .bss
    x              resb  10
    a              resb  10
    result         resb  10

section .text
    global _start

_start:
    ; Запрашиваем значение переменной x

```

```

mov     eax, msg_input_x
call   sprint
; Организует ввод с клавиатуры значения переменной a
mov     ecx, x
mov     edx, 10
call   sread
; Преобразует строку, полученную с клавиатуры, в число
mov     eax, x
call   atoi
mov     [x], eax

; Запрашиваем значение переменной a
mov     eax, msg_input_a
call   sprint
; Организует ввод с клавиатуры значения переменной b
mov     ecx, a
mov     edx, 10
call   sread
; Преобразует строку, полученную с клавиатуры, в число
mov     eax, a
call   atoi
mov     [a], eax

; Сравниваем a и 7
mov     ecx, [a]
cmp     ecx, 7
jl     a_lower_7
; Если a >= 7, то result = a - 7
mov     eax, [a]

```

```
sub    eax, 7
mov    [result], eax
jmp   fin
```

a\_lower\_7:

*; Если  $a < 7$ , то  $result = a * x$*

```
mov    eax, [a]
mov    ecx, [x]
mul    ecx
mov    [result], eax
```

fin:

*; Выводим сообщение пользователю*

```
mov    eax, msg_result
```

```
call sprint
```

*; Печатаем результат*

```
mov    eax, [result]
```

```
call iprintLF
```

```
call quit
```