

# **Лабораторная работа №7**

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

Турсунбоев Сардорбек

# Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	23

## Список иллюстраций

3.1	Код программы lab7-1.asm . . . . .	8
3.2	Тестирование программы lab7-1.asm . . . . .	9
3.3	Код программы lab7-1.asm . . . . .	10
3.4	Тестирование программы lab7-1.asm . . . . .	11
3.5	Код программы lab7-1.asm . . . . .	12
3.6	Тестирование программы lab7-1.asm . . . . .	13
3.7	Код программы lab7-2.asm . . . . .	14
3.8	Тестирование программы lab7-2.asm . . . . .	15
3.9	Файл листинга lab7-2 . . . . .	16
3.10	Ошибка трансляции lab7-2 . . . . .	17
3.11	Файл листинга с ошибкой lab7-2 . . . . .	18
3.12	Код программы lab7-3.asm . . . . .	19
3.13	Тестирование программы lab7-3.asm . . . . .	20
3.14	Код программы lab7-4.asm . . . . .	21
3.15	Тестирование программы lab7-4.asm . . . . .	22

## **Список таблиц**

# 1 Цель работы

Целью 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. Изучить команды сравнения
2. Изучить команды переходов
3. Рассмотреть примеры программ
4. Изучить файл листинга
5. Выполнить самостоятельное задание

## 3 Выполнение лабораторной работы

Создал каталог для программ лабораторной работы № 7 и файл lab7-1.asm.

В NASM инструкция `jmp` используется для осуществления безусловных переходов. Давайте рассмотрим пример программы, где используется инструкция `jmp`.

Написал текст программы из листинга 7.1 в файл 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. 3.1: Код программы lab7-1.asm

Создал исполняемый файл и запустил его.



```
[sktursunboev@fedora lab07]$ touch lab7-1.asm
[sktursunboev@fedora lab07]$ nasm -f elf lab7-1.asm
[sktursunboev@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[sktursunboev@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[sktursunboev@fedora lab07]$
```

Рис. 3.2: Тестирование программы lab7-1.asm

Инструкция `jmp` позволяет осуществлять переходы не только вперед, но и назад. Мы изменим программу так, чтобы она сначала выводила “Сообщение № 2”, затем “Сообщение № 1” и завершала работу. Для этого мы добавим в текст программы после вывода “Сообщения № 2” инструкцию `jmp` с меткой `_label1` (переход к инструкциям вывода “Сообщения № 1”) и после вывода “Сообщения № 1” добавим инструкцию `jmp` с меткой `_end` (переход к инструкции `call quit`).

Изменил текст программы в соответствии с листингом 7.2.

Открыть ▾

+

lab7-1.asm  
~/work/lab07

```
%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. 3.3: Код программы lab7-1.asm

```
[sktursunboev@fedora lab07]$  
[sktursunboev@fedora lab07]$ touch lab7-1.asm  
[sktursunboev@fedora lab07]$ nasm -f elf lab7-1.asm  
[sktursunboev@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1  
[sktursunboev@fedora lab07]$ ./lab7-1  
Сообщение № 2  
Сообщение № 3  
[sktursunboev@fedora lab07]$ nasm -f elf lab7-1.asm  
[sktursunboev@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1  
[sktursunboev@fedora lab07]$ ./lab7-1  
Сообщение № 2  
Сообщение № 1  
[sktursunboev@fedora lab07]$
```


Рис. 3.4: Тестирование программы lab7-1.asm

Изменил текст программы, изменив инструкции `jmp`, чтобы вывод программы был следующим:

Сообщение № 3

Сообщение № 2

Сообщение № 1

Открыть ▾ 

lab7-1.asm  
~/work/lab07

```
%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. 3.5: Код программы lab7-1.asm

```

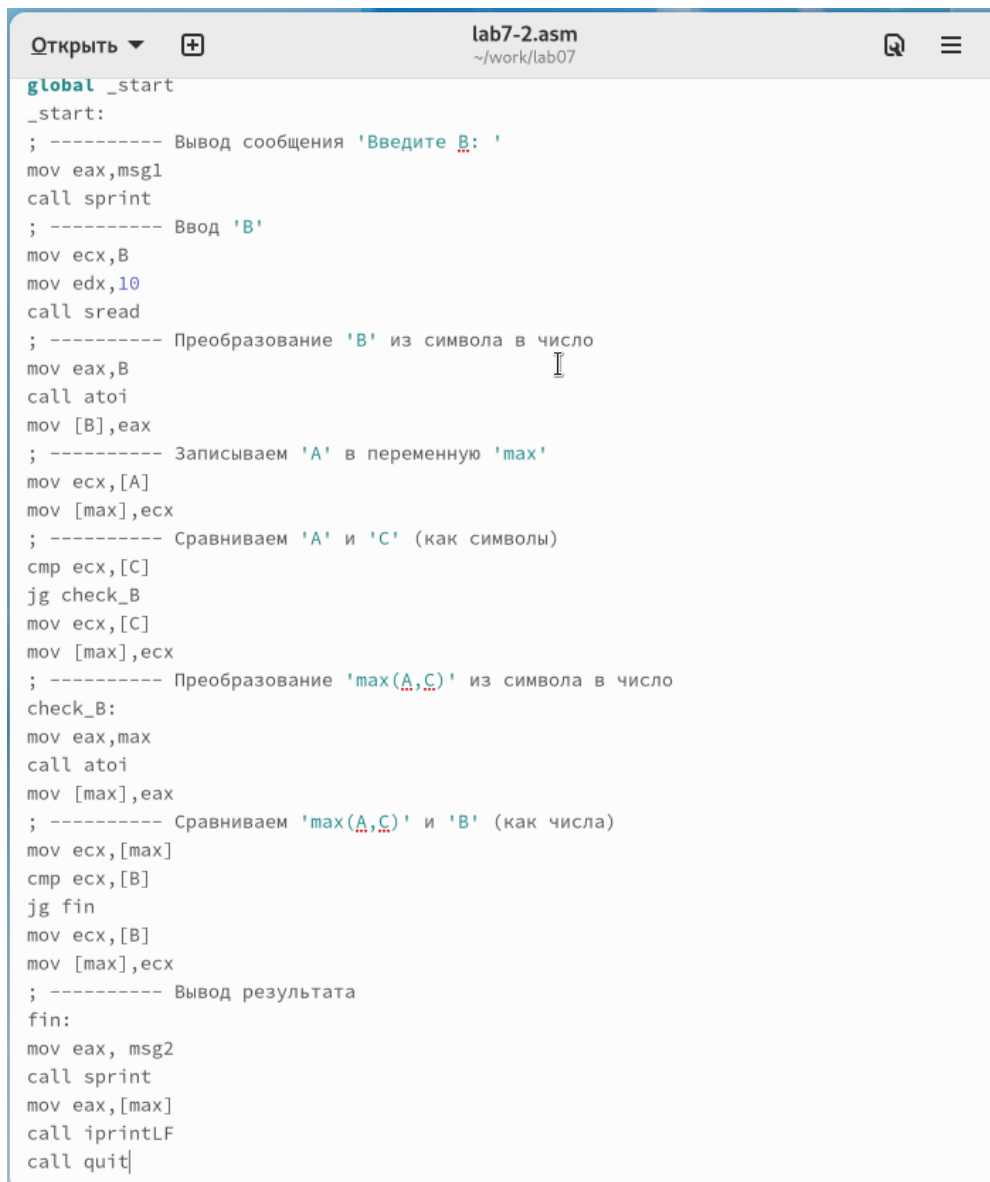
[sktursunboev@fedora lab07]$ touch lab7-1.asm
[sktursunboev@fedora lab07]$ nasm -f elf lab7-1.asm
[sktursunboev@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[sktursunboev@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[sktursunboev@fedora lab07]$ nasm -f elf lab7-1.asm
[sktursunboev@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[sktursunboev@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 1
[sktursunboev@fedora lab07]$ nasm -f elf lab7-1.asm
[sktursunboev@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[sktursunboev@fedora lab07]$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[sktursunboev@fedora lab07]$ █

```

Рис. 3.6: Тестирование программы lab7-1.asm

Использование инструкции `jmp` приводит к переходу в любом случае. Однако, при написании программ часто необходимо использовать условные переходы, то есть переход должен происходить, если выполнено определенное условие. Давайте рассмотрим программу, которая определяет и выводит на экран наибольшую из трех целочисленных переменных: `A`, `B` и `C`. Значения для `A` и `C` задаются в программе, а значение `B` вводится с клавиатуры.

Создал исполняемый файл и проверил его работу для разных значений `B`.



```
Открыть + lab7-2.asm ~/work/lab07
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A]
mov [max],ecx
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi
mov [max],eax
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint
mov eax,[max]
call iprintLF
call quit
```

Рис. 3.7: Код программы lab7-2.asm

```
[sktursunboev@fedora lab07]$ touch lab7-2.asm
[sktursunboev@fedora lab07]$ nasm -f elf lab7-2.asm
[sktursunboev@fedora lab07]$ ld -m elf_i386 lab7-2.o -o lab7-2
[sktursunboev@fedora lab07]$ ./lab7-2
Введите B: 10
Наибольшее число: 50
[sktursunboev@fedora lab07]$ ./lab7-2
Введите B: 40
Наибольшее число: 50
[sktursunboev@fedora lab07]$ ./lab7-2 60
Введите B: 60
Наибольшее число: 60
[sktursunboev@fedora lab07]$
```

Рис. 3.8: Тестирование программы lab7-2.asm

Обычно при ассемблировании с помощью `nasm` создается только объектный файл. Для получения файла листинга можно использовать ключ `-l` и указать имя файла листинга в командной строке.

```

Открыть  + lab7-2.lst ~\work\lab07 Стр. 1, Поз. 1
160 160 000000D8 59 <1> pop ecx
161 161 000000D9 5B <1> pop ebx
162 162 000000DA C3 <1> ret
163 163 <1>
164 164 <1>
165 165 <1> ;----- quit -----
166 166 <1> ; функция завершения программы
167 167 <1> quit:
168 168 000000DB BB00000000 <1> mov ebx, 0
169 169 000000E0 B801000000 <1> mov eax, 1
170 170 000000E5 CD80 <1> int 80h
171 171 000000E7 C3 <1> ret
172 2 section .data
173 3 00000000 D092D0B2D0B5D0B4D0- msg1 db 'Введите B: ',0h
174 3 00000009 B8D182D0B520423A20-
175 3 00000012 00
176 4 00000013 D09DD0B0D0B8D0B1D0- msg2 db "Наибольшее число: ",0h
177 4 0000001C BED0BBD18CD188D0B5-
178 4 00000025 D0B520D187D0B8D181-
179 4 0000002E D0BBD0BE3A2000
180 5 00000035 32300000 A dd '20'
181 6 00000039 35300000 C dd '50'
182 7 section .bss
183 8 00000000 <res Ah> max resb 10
184 9 0000000A <res Ah> B resb 10
185 10 section .text
186 11 global _start
187 12 _start:
188 13 ; ----- Вывод сообщения 'Введите B: '
189 14 000000E8 B8[00000000] mov eax,msg1
190 15 000000ED E810FFFFFF call sprint
191 16 ; ----- Ввод 'B'
192 17 000000F2 B9[0A000000] mov ecx,B
193 18 000000F7 BA0A000000 mov edx,10

```

Рис. 3.9: Файл листинга lab7-2

Я внимательно ознакомился с форматом и содержимым файла листинга. Подробно поясню некоторые элементы листинга.

#### строка 168

- 168 - номер строки
- 000000DB - адрес
- BB00000000 - машинный код
- mov ebx, 0 - Помещаем 0 в регистр ebx

#### строка 169

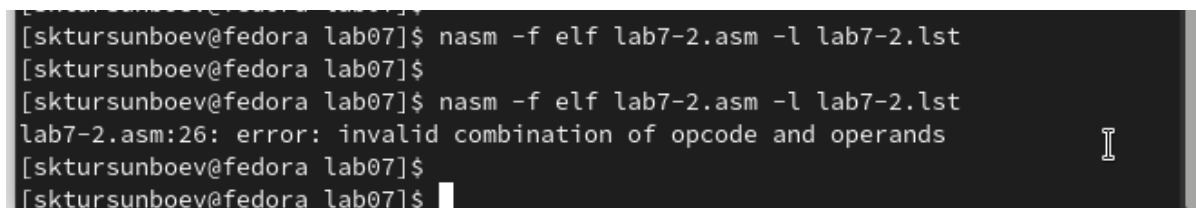


- 169 - номер строки
- 000000E0 - адрес
- B801000000 - машинный код
- mov eax, 1 - Помещаем 1 в регистр eax

#### строка 170

- 26 - номер строки
- 000000E5 - адрес
- CD80 - машинный код
- int 80h - вызов ядра

Открыл файл программы lab7-2.asm и в инструкции с двумя операндами удалил один из операндов. После этого я выполнил трансляцию программы и получил файл листинга.



```
[sktursunboev@fedora lab07]$ nasm -f elf lab7-2.asm -l lab7-2.lst
[sktursunboev@fedora lab07]$
[sktursunboev@fedora lab07]$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:26: error: invalid combination of opcode and operands
[sktursunboev@fedora lab07]$
[sktursunboev@fedora lab07]$
```

Рис. 3.10: Ошибка трансляции lab7-2

```

191 16 ; ----- Ввод 'B'
192 17 000000F2 B9[0A000000] mov ecx,B
193 18 000000F7 BA0A000000 mov edx,10
194 19 000000FC E842FFFFFF call sread
195 20 ; ----- Преобразование 'B' из символа в число
196 21 00000101 B8[0A000000] mov eax,B
197 22 00000106 E801FFFFFF call atoi
198 23 0000010B A3[0A000000] mov [B],eax
199 24 ; ----- Записываем 'A' в переменную 'max'
200 25 00000110 8B0D[35000000] mov ecx,[A]
201 26 mov [max],
202 26 ***** error: invalid combination of opcode and operands
203 27 ; ----- Сравниваем 'A' и 'C' (как символы)
204 28 00000116 3B0D[39000000] cmp ecx,[C]
205 29 0000011C 7F0C ig check B
206 30 0000011E 8B0D[39000000] mov ecx,[C]
207 31 00000124 890D[00000000] mov [max],ecx
208 32 ; ----- Преобразование 'max(A,C)' из символа в число
209 33 check B:
210 34 0000012A B8[00000000] mov eax,max
211 35 0000012F E868FFFFFF call atoi
212 36 00000134 A3[00000000] mov [max],eax
213 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
214 38 00000139 8B0D[00000000] mov ecx,[max]
215 39 0000013F 3B0D[0A000000] cmp ecx,[B]
216 40 00000145 7F0C ig fin
217 41 00000147 8B0D[0A000000] mov ecx,[B]
218 42 0000014D 890D[00000000] mov [max],ecx
219 43 ; ----- Вывод результата
220 44 fin:
221 45 00000153 B8[13000000] mov eax, msg2
222 46 00000158 E8B2FFFFFF call sprint
223 47 0000015D A1[00000000] mov eax,[max]
224 48 00000162 E81FFFFFFF call iprintLF
225 49 00000167 E86FFFFFFF call quit

```

Рис. 3.11: Файл листинга с ошибкой lab7-2

Хотя объектный файл не удалось создать из-за ошибки, я все же получил листинг, в котором указано место ошибки.

Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу

для варианта 4 - 8, 88, 68

```
Открыть ▾ + lab7-3.asm
~/work/lab07
28 mov ecx,msgC
29 call sprint
30 mov ecx,B
31 mov edx,80
32 call sread
33 mov eax,B
34 call atoi
35 mov [B],eax
36
37 mov eax,msgC
38 call sprint
39 mov ecx,C
40 mov edx,80
41 call sread
42 mov eax,C
43 call atoi
44 mov [C],eax
45
46 mov ecx,[A]
47 mov [min],ecx
48
49 cmp ecx, [B]
50 jl check_C
51 mov ecx, [B]
52 mov [min], ecx
53
54 check_C:
55 cmp ecx, [C]
56 jl finish
57 mov ecx,[C]
58 mov [min],ecx
59
60 finish:
61 mov eax,answer
62 call sprint
63
64 mov eax, [min]
65 call iprintLF
66
67 call quit
```

Рис. 3.12: Код программы lab7-3.asm


```
[sktursunboev@fedora lab07]$ nasm -f elf lab7-3.asm
[sktursunboev@fedora lab07]$ ld -m elf_i386 lab7-3.o -o lab7-3
[sktursunboev@fedora lab07]$ ./lab7-3
Input A: 8
Input B: 88
Input C: 68
Smallest: 8
[sktursunboev@fedora lab07]$
```

Рис. 3.13: Тестирование программы lab7-3.asm

Напишите программу, которая для введенных с клавиатуры значений  $x$  и  $a$  вычисляет значение заданной функции  $f(x)$  и выводит результат вычислений. Вид функции  $f(x)$  выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений  $X$  и  $a$  из 7.6.

для варианта 4

$$\begin{cases} 2 * x + a, a \neq 0 \\ 2 * x + 1, a = 0 \end{cases}$$

Открыть ▾ 

lab7-4.asm  
~/work/lab07

```
14 _start:
15     mov eax,msgA
16     call sprint
17     mov ecx,A
18     mov edx,80
19     call sread
20     mov eax,A
21     call atoi
22     mov [A],eax
23
24     mov eax,msgX
25     call sprint
26     mov ecx,X
27     mov edx,80
28     call sread
29     mov eax,X
30     call atoi
31     mov [X],eax
32
33     mov edx, [A]
34     mov ebx, 0
35     cmp edx, ebx
36     jne first
37     jmp second
38
39 first:
40     mov eax,[X]
41     mov ebx, 2
42     mul ebx
43     add eax,[A]
44     call iprintLF
45     call quit
46 second:
47     mov eax,[X]
48     mov ebx,2
49     mul ebx
50     add eax,1
51     call iprintLF
52     call quit
```

Рис. 3.14: Код программы lab7-4.asm

```
[sktursunboev@fedora lab07]$ touch lab7-4.asm
[sktursunboev@fedora lab07]$ nasm -f elf lab7-4.asm
[sktursunboev@fedora lab07]$ ld -m elf_i386 lab7-4.o -o lab7-4
[sktursunboev@fedora lab07]$ ./lab7-4
Input A: 0
Input X: 3
7
[sktursunboev@fedora lab07]$ ./lab7-4
Input A: 2
Input X: 3
8
[sktursunboev@fedora lab07]$
```

Рис. 3.15: Тестирование программы lab7-4.asm

## 4 Выводы

Изучили команды условного и безусловного переходов, познакомились с фалом листинга.