

# **Отчёт по лабораторной работе №7**

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

Малкина Дарья Александровна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Реализация переходов в NASM . . . . .	6
2.2	Изучение структуры файлы листинга . . . . .	9
<b>3</b>	<b>Выполнение задания для самостоятельной работы</b>	<b>11</b>
<b>4</b>	<b>Выводы</b>	<b>16</b>

# Список иллюстраций

2.1	Программа lab7-1 . . . . .	6
2.2	Редактирование текста программы lab7-1 . . . . .	7
2.3	Измененная программа lab7-1 . . . . .	7
2.4	Редактирование текста программы lab7-1 . . . . .	8
2.5	Измененная программа lab7-1 . . . . .	8
2.6	Программа lab7-2 . . . . .	9
2.7	Создание листинга для lab7-2 . . . . .	9
2.8	Листинг программы lab7-2 . . . . .	9
2.9	Удаление операнда . . . . .	10
2.10	Ошибка в листинге . . . . .	10
3.1	A, B, C, min . . . . .	11
3.2	Преобразование 'B' . . . . .	12
3.3	Сравнение 'A' и 'C' . . . . .	12
3.4	Сравнение 'min' и 'B' . . . . .	13
3.5	Результат . . . . .	13
3.6	Результат . . . . .	14
3.7	Ввод переменных и преобразование X и A в числа . . . . .	14
3.8	Вычисление . . . . .	15
3.9	Результат . . . . .	15

## **Список таблиц**

# 1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

## 2 Выполнение лабораторной работы

### 2.1 Реализация переходов в NASM

1. Создадим каталог для программ лабораторной работы №7, в нем создадим файл lab7-1.asm и введем в созданный файл текст программы из листинга 7.1. После создаем исполняемый файл и запускаем его:

```
[damalkina@ArchVBox lab07]$ ls
in_out.asm  lab7-1.asm
[damalkina@ArchVBox lab07]$ nasm -f elf lab7-1.asm -o lab7-1.o
[damalkina@ArchVBox lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[damalkina@ArchVBox lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[damalkina@ArchVBox lab07]$
```

Рис. 2.1: Программа lab7-1

2. Изменим программу, чтобы она выводила сначала 'Сообщение №2', потом 'Сообщение №1' и завершала работу:

```

10 _start:
11
12 jmp _label2
13
14 _label1:
15 mov eax, msg1 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 1'
17 jmp _end
18
19 _label2:
20 mov eax, msg2 ; Вывод на экран строки
21 call sprintf ; 'Сообщение № 2'
22 jmp _label1

```

Рис. 2.2: Редактирование текста программы lab7-1

Создаем исполняемый файл и запускаем его:

```

[damalkina@ArchVBox lab07]$ nasm -f elf lab7-1.asm -o lab7-1.o
[damalkina@ArchVBox lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[damalkina@ArchVBox lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 1
[damalkina@ArchVBox lab07]$

```

Рис. 2.3: Измененная программа lab7-1

Снова изменим программу, чтобы теперь она выводила сначала 'Сообщение №3', потом 'Сообщение №2', затем 'Сообщение №3' и завершала работу:

```

12 jmp _label3
13
14 _label1:
15 mov eax, msg1 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 1'
17 jmp _end
18
19 _label2:
20 mov eax, msg2 ; Вывод на экран строки
21 call sprintf ; 'Сообщение № 2'
22 jmp _label1
23
24 _label3:
25 mov eax, msg3 ; Вывод на экран строки
26 call sprintf ; 'Сообщение № 3'
27 jmp _label2

```

Рис. 2.4: Редактирование текста программы lab7-1

Создаем исполняемый файл и запускаем его:

```

[damalkina@ArchVBox lab07]$ nasm -f elf lab7-1.asm -o lab7-1.o
[damalkina@ArchVBox lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[damalkina@ArchVBox lab07]$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[damalkina@ArchVBox lab07]$

```

Рис. 2.5: Измененная программа lab7-1

3. Создадим файл lab7-2.asm и введем в созданный файл текст программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А,В и С. После создаем исполняемый файл и запускаем его:



```

[damalkina@ArchVBox lab07]$ touch lab7-2.asm
[damalkina@ArchVBox lab07]$ nasm -f elf lab7-2.asm -o lab7-2.o
[damalkina@ArchVBox lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[damalkina@ArchVBox lab07]$ ./lab7-2
Введите B: 10
Наибольшее число: 50
[damalkina@ArchVBox lab07]$ ./lab7-2
Введите B: 100
Наибольшее число: 100
[damalkina@ArchVBox lab07]$ ./lab7-2
Введите B: 43
Наибольшее число: 50
[damalkina@ArchVBox lab07]$ ./lab7-2
Введите B: 67
Наибольшее число: 67
[damalkina@ArchVBox lab07]$

```

Рис. 2.6: Программа lab7-2

## 2.2 Изучение структуры файлы листинга

1. Создадим файл листинга для программы из файла lab7-2.asm:

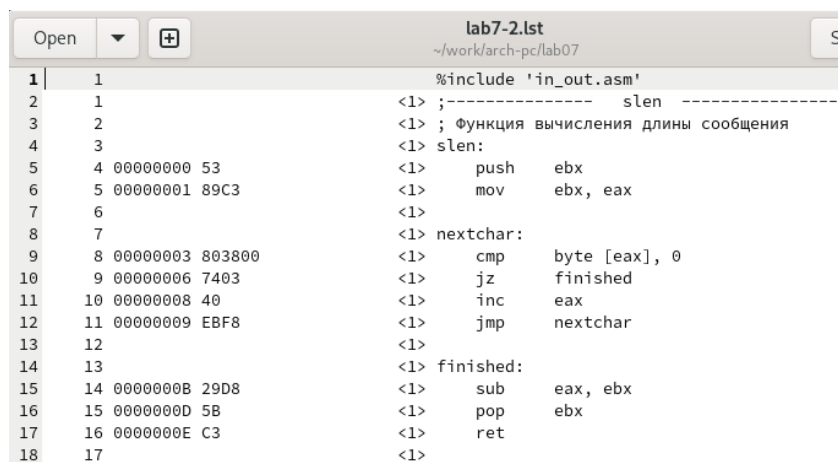
```

[damalkina@ArchVBox lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
[damalkina@ArchVBox lab07]$ ls
in_out.asm  lab7-1.asm  lab7-2.lst
lab7-1     lab7-1.o   lab7-2.asm  lab7-2.o

```

Рис. 2.7: Создание листинга для lab7-2

Откроем его с помощью текстового редактора:



```

lab7-2.lst
~/work/arch-pc/lab07

1 | 1 | %include 'in_out.asm'
2 | 1 | <1> ;----- slen -----
3 | 2 | <1> ; функция вычисления длины сообщения
4 | 3 | <1> slen:
5 | 4 | 00000000 53 | <1> push ebx
6 | 5 | 00000001 89C3 | <1> mov ebx, eax
7 | 6 | | <1>
8 | 7 | | <1> nextchar:
9 | 8 | 00000003 803800 | <1> cmp byte [eax], 0
10 | 9 | 00000006 7403 | <1> jz finished
11 | 10 | 00000008 40 | <1> inc eax
12 | 11 | 00000009 EBF8 | <1> jmp nextchar
13 | 12 | | <1>
14 | 13 | | <1> finished:
15 | 14 | 0000000B 29D8 | <1> sub eax, ebx
16 | 15 | 0000000D 5B | <1> pop ebx
17 | 16 | 0000000E C3 | <1> ret
18 | 17 | | <1>

```

Рис. 2.8: Листинг программы lab7-2

## 2. Рассмотрим 28, 34 и 51 строки листинга:

28 00000101 B8[0A000000] mov eax,B 28 - номер строки в листинге 00000101 - адрес в памяти, с которого начинается команда B8 - машинный код для команды mov eax шестнадцетиричном представлении, команда копирует значение B в eax [0A000000] - адрес в памяти, по которому хранится значение переменной B mov eax,B - исходный текст программы

34 00000116 890D[00000000] mov [max],ecx 34 - номер строки 00000116 - адрес в памяти, с которого начинается команда 890D - машинный код для команды mov [max],ecx в шестнадцетиричном представлении, команда копирует значение из регистра ecx в переменную [max] [00000000] - адрес в памяти, по которому хранится значение переменной [max] mov [max],ecx - исходный текст программы

51 0000014B 7F0C jg fn 51 - номер строки в листинге 0000014B - адрес в памяти, с которого начинается программа 7F0C - машинный код для команды jg fn в шестнадцатеричном представлении, команда выполняет условный переход на метку fn, если значение в регистре flags указывают на то, что результат сравнения max(A,C)>B истинно jg fn - исходный текст программы

## 3. Вернемся к файлу lab7-2.asm и изменим текст программы, удалим операнд B в инструкции mov eax,B на 28 строке кода:

```
27 ; ----- Преобразование 'B' из символа в число
28 mov eax
29 call atoi ; Вызов подпрограммы перевода символа в число
30 mov [B],eax ; запись преобразованного числа в 'B'
```

Рис. 2.9: Удаление операнда

После изменений в тексте программы выполняем трансляцию с получением файла листинга, заметим, что теперь в 28 строке сообщение об ошибке:

```
26
27 ; ----- Преобразование 'B' из символа в число
28 mov eax
28 ***** error: invalid combination of opcode and operands
29 00000101 E896FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
30 00000106 A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
31
```

Рис. 2.10: Ошибка в листинге

### 3 Выполнение задания для самостоятельной работы

1. Напишем программу нахождения наименьшей из 3 целочисленных переменных A, B и C, сначала задаем переменные и выделяем память для результата:

```
1 %include 'in_out.asm'
2 section .data
3
4 msg2 db 'Наименьшее число: ',0h
5 A dd '62'
6 B dd '54'
7 C dd '87'
8
9 section .bss
10 min resb 10
11
12 section .text
13 global _start
14
15 _start:
```

Рис. 3.1: A, B, C, min

Преобразовываем 'B' из символа в число:

```

15 _start:
16
17 ; ----- Преобразование 'B' из символа в число
18 mov eax,B
19 call atoi ; Вызов подпрограммы перевода символа в число
20 mov [B],eax ; запись преобразованного числа в 'B'

```

Рис. 3.2: Преобразование 'B'

Далее записываем 'A' в переменную min, после чего сравниваем 'A' и 'C' как символы, если 'A' будет меньше 'C', то по флагу jб check\_B мы перейдем к сравнению 'A' и 'B', иначе будем сравнивать 'C' и 'B':

```

22 ; ----- Записываем 'A' в переменную 'min'
23 mov ecx,[A] ; 'ecx = A'
24 mov [min],ecx ; 'min = A'
25
26 ; ----- Сравниваем 'A' и 'C' (как символы)
27 cmp ecx,[C] ; Сравниваем 'A' и 'C'
28 jб check_B ; если 'A<C', то переход на метку 'check_B'
29 mov ecx,[C] ; иначе 'ecx = C'
30 mov [min],ecx ; 'min = C'

```

Рис. 3.3: Сравнение 'A' и 'C'

Если 'A' меньше 'C', то переходим к сравнению 'A' и 'B', для этого преобразовываем 'min' из символа в число, и после сравниваем 'min' и 'B' как числа, в конце выводим результат:

```

32 ; ----- Преобразование 'min(A,C)' из символа в число
33 check_B:
34 mov eax,min
35 call atoi ; Перевод символа в число
36 mov [min],eax ; запись преобразованного числа в 'min'
37
38 ; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
39 mov ecx,[min]
40 cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
41 jb fin ; если 'min(A,C)<B', то переход на 'fin'
42 mov ecx,[B] ; иначе 'ecx = B'
43 mov [min],ecx
44
45 ; ----- Вывод результата
46 fin:
47 mov eax, msg2
48 call sprint ; Вывод сообщения
49 mov eax,[min]
50 call iprintLF ; Вывод 'min(A,B,C)'
51 call quit ; Выход

```

Рис. 3.4: Сравнение 'min' и 'B'

Проверяем результат:

```

[damalkina@ArchVBox lab07]$ nasm -f elf lab7-var5-1.asm -o lab7-var5-1.o
[damalkina@ArchVBox lab07]$ ld -m elf_i386 -o lab7-var5-1 lab7-var5-1.o
[damalkina@ArchVBox lab07]$ ./lab7-var5-1
Наименьшее число: 54

```

Рис. 3.5: Результат

2. Напишем программу, которая для введенных с клавиатуры значений X и A вычисляет значение функции  $f(x)=15$  при  $A>X$ ,  $f(x)=2*(X-A)$  при  $A\leq X$  и выводит результат вычислений:

```

1 %include 'in_out.asm'
2 section .data
3
4 msg1 db 'Введите число X: ',0h
5 msg2 db 'Введите число A: ',0h
6 msg3 db 'Значение функции f(x) = ',0h
7
8 section .bss
9 X resb 10
10 A resb 10
11 F resb 10

```

Рис. 3.6: Результат

Запишем команды для вывода сообщений и ввода переменных с клавиатуры, преобразуем переменные из символов в числа:

```

16 _start:
17
18 ; ----- Вывод сообщения 'Введите X: '
19 mov eax,msg1
20 call sprint
21 ; ----- Ввод 'X'
22 mov ecx,X
23 mov edx,10
24 call sread
25 ; ----- Преобразование 'X' из символа в число
26 mov eax,X
27 call atoi
28 mov [X],eax ; запись преобразованного числа в 'X'
29
30 ; ----- Вывод сообщения 'Введите A: '
31 mov eax,msg2
32 call sprint
33 ; ----- Ввод 'A'
34 mov ecx,A
35 mov edx,10
36 call sread
37 ; ----- Преобразование 'A' из символа в число
38 mov eax,A
39 call atoi
40 mov [A],eax ; запись преобразованного числа в 'A'

```

Рис. 3.7: Ввод переменных и преобразование X и A в числа

Теперь переменные можно сравнить как числа, если A будет юольше X, то по флагу jg com1 мы перейдем к сравнению вычислению выражения  $2*(X-A)$ , иначе функция примет значение равное 15:

```
42 ; ----- Сравниваем 'X' и 'A' (как числа)
43 mov eax,[X]
44 cmp eax,[A] ; Сравниваем 'X' и 'A'
45 jg com1 ; если 'X>A', то переход на 'com1'
46
47 ; ----- если 'X <= A'
48 mov dword [F],15 ; 'F(x)=15'
49 jmp fin
50
51 ; ----- если 'X > A'
52 com1:
53 sub eax,[A] ; разность (x-a)
54 add eax,eax ; умножаем на 2
55 mov [F],eax
56
57 ; ----- Вывод результата
58 fin:
59 mov eax, msg3
60 call sprint ; Вывод сообщения
61 mov eax,[F]
62 call iprintLF ; Вывод
63 call quit ; Выход
```

Matlab ▾ Tab Width

Рис. 3.8: Вычисление

Проверим результат:

```
[damalkina@ArchVBox lab07]$ ./lab7-var5-2
Введите число X: 1
Введите число A: 2
Значение функции f(x) = 15
[damalkina@ArchVBox lab07]$ ./lab7-var5-2
Введите число X: 2
Введите число A: 1
Значение функции f(x) = 2
[damalkina@ArchVBox lab07]$
```

Рис. 3.9: Результат

## **4 Выводы**

В ходе выполнения лабораторной работы мы изучили команды условного и безусловного переходов. А также написали программы с использованием переходов. Мы познакомились с назначением и структурой файла листинга.