

# **Лабораторная работа №7**

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

Краснопер Данила Олегович

# Содержание

1	Цель работы	3
2	Выполнение лабораторной работы	4
3	Самостоятельная работа.	10
4	Вывод	15

# 1 Цель работы

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

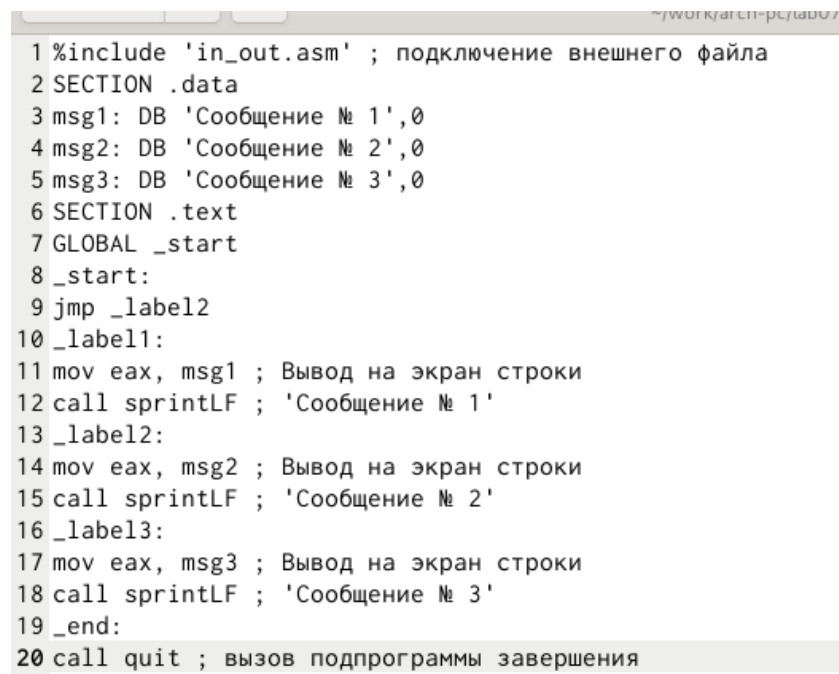
## 2 Выполнение лабораторной работы

1) Я создал каталог lab7 и внутри создал файл lab7-1.asm

```
dokrasnoper@dk3n51 ~ $ mkdir ~/work/arch-pc/lab07
dokrasnoper@dk3n51 ~ $ cd ~/work/arch-pc/lab07
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ touch lab7-1.asm
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ ls
lab7-1.asm
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $
```

Рис. 2.1: Создание файла lab8-1.asm

2) Я ввел в файл текст программы и запустил его.



```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintLF ; 'Сообщение № 1'
13 _label2:
14 mov eax, msg2 ; Вывод на экран строки
15 call sprintLF ; 'Сообщение № 2'
16 _label3:
17 mov eax, msg3 ; Вывод на экран строки
18 call sprintLF ; 'Сообщение № 3'
19 _end:
20 call quit ; вызов подпрограммы завершения
```

Рис. 2.2: Текст в файле lab8-1.asm

3) Я создал исполняемый файл и запустил его. Результат соответствовал нужному.

```
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $
```

Рис. 2.3: Запуск программы lab8-1

4) Я изменил текст программы чтобы выводился нужный ответ и создал исполняемый файл.

```
lab7-1.asm [----] 0 L:[ 1+14 15/ 23] *(377 / 671b) 0109 0x06
#include 'in_out.asm' ; подключение внешнего файла
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 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 2.4: Изменение текста

```
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $
```

Рис. 2.5: Проверка работы программы

5) Я изменил текст программы чтобы сначала выводило сообщение 3, затем 2, затем 1.

```
lab7-1.asm      [-M--]  0 L:[ 3+21 24/ 24] *(683 / 68
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 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения
□
```

Рис. 2.6: Изменение текста

6) Запустил программу и проверил ее работу.

```
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ □
```

Рис. 2.7: Запуск программы

7) Я создал файл lab7-2.asm и написал текст программы.

```

lab7-2.asm      [----] 17 L:[ 10+25 35/ 49] *(1117/1743b) 0032 0x020  [*][
section .text
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 ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход

```

Рис. 2.8: Текст программы для сравнения чисел

8) Я ввел два разных числа чтобы проверить как работает программа.

```

dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 78
Наибольшее число: 78
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 14
Наибольшее число: 50
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ 

```

Рис. 2.9: Программа для сравнения чисел

9) Я создал файл листинга lab7-2.lst и открыл его.

```
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ mcedit lab7-2.lst
```

Рис. 2.10: Файл листинга lab8-2.lst

- 10) Проанализировав файл, я понял как он работает и какие значения выводит.
- 11) Эта строка находится на 21 месте, ее адрес “00000101”, Машинный код - B8A000000, а `mov eax,B` - исходный текст программы, означающий что в регистр `eax` мы вносим значения переменной `B`.

```
21 00000101 B8[0A000000]          mov eax,B
```

Рис. 2.11: Объяснения первой строки

- 2) Эта строка находится на 35 месте, ее адрес “00000135”, Машинный код - E862FFFFFF, а `call atoi` - исходный текст программы, означающий что символ лежащий в строке выше переводится в число.

```
35 00000135 E862FFFFFF          call atoi ;
```

Рис. 2.12: Объяснения второй строки

- 3) Эта строка находится на 47 месте, ее адрес “00000163”, Машинный код - A100000000, а `mov eax,тах` - исходный текст программы, означающий что число хранившееся в переменной `тах` записывается в регистр `eax`.

```
47 00000163 A1[00000000]          mov eax,[тах]
```

Рис. 2.13: Объяснения третьей строки

- 11) В строке `mov eax,тах` я убрал `тах` и попробовал создать файл. Должно было выдать ошибку, так как для программы нужно два операнда. Но исходный файл автоматически исправлялся и прервать это было не возможно, из-за чего терминал ошибку не выдал.



```
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $
```

Рис. 2.14: Создание файла без одного операнда

### **3 Самостоятельная работа.**

- 1) Я написал программу для нахождения меньшего из трех чисел. Для большего удобства я сделал ввод чисел с клавиатуры. У меня девятый вариант, поэтому числа были :24,98,15. Программа вывела меньшее из этих чисел.

```

1 %include 'in_out.asm'
2
3 SECTION .data
4 A1 DB 'Введите число A: ',0h
5 B1 DB 'Введите число B: ',0h
6 C1 DB 'Введите число C: ',0h
7 otv DB 'Наименьшее число: ',0h
8 SECTION .bss
9 min RESB 20
10 A RESB 20
11 B RESB 20
12 C RESB 20
13
14 SECTION .text
15 GLOBAL _start
16 _start:
17
18 mov eax,A1
19 call sprint
20
21 mov ecx,A
22 mov edx,20
23 call sread
24
25 mov eax, A
26 call atoi
27 mov [A],eax
28
29 xor eax,eax
30
31 mov eax,B1
32 call sprint
33
34 mov ecx,B
35 mov edx,20
36 call sread
37
38 mov eax,B
39 call atoi
40 mov [B],eax
41
42 xor eax,eax
43
44 mov ecx, [A]
45 mov [min],ecx
46 mov ecx,[min]
47

```

Рис. 3.1: Текст программы

```
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ ./lab7-3
Введите число A: 24
Введите число B: 98
Введите число C: 15
Наименьшее число: 15
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ █
```

Рис. 3.2: Результат работы программы

- 2) Я написал программу, чтобы она вычисляла выражение при введенных X и A. Так как у меня 9 вариант, то программа написана для 9 варианта.

```
lab7-4.asm [
call atoi

mov [A],eax
mov eax,[A]
mov ebx,[X]
mov [X],ebx
cmp ebx,eax
jle com1
mov[F],eax
jmp fin
com1:
add eax,ebx
mov[F],eax

fin:
mov eax, msg3
call sprint
```

Рис. 3.3: Текст программы

```
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ ./lab7-4
Введите число X: 5
Введите число A: 7
Значение функции  $f(x) = 12$ 
dokrasnoper@dk3n51 ~/work/arch-pc/lab07 $ ./lab7-4
Введите число X: 6
Введите число A: 4
Значение функции  $f(x) = 4$ 
```

Рис. 3.4: Проверка работы программы

## 4 Вывод

Я изучил команды условного и безусловного перехода. Приобрел навыки написания программ с переходами.