

# **Отчёт по лабораторной работе №7**

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

Мария Валерьевна Грачева

# Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Выполнение самостоятельной работы	13
5	Выводы	18
	Список литературы	19

## Список иллюстраций

3.1	Создание каталога и файла lab7-1.asm . . . . .	7
3.2	Листинг программы . . . . .	7
3.3	Запуск файла . . . . .	8
3.4	Изменение текста программы . . . . .	8
3.5	Запуск файла 2 . . . . .	8
3.6	Новый листинг программы . . . . .	9
3.7	Запуск файла 3 . . . . .	9
3.8	Создание каталога и файла lab7-2.asm . . . . .	10
3.9	Листинг программы 2 . . . . .	10
3.10	Проверка работы файла . . . . .	11
3.11	Проверка работы файла 2 . . . . .	11
3.12	Команда nasm с ключом -l и mcedit . . . . .	11
3.13	Открытый файл lst . . . . .	12
4.1	Создание файла для сам работы . . . . .	13
4.2	Введение текста в сам работу . . . . .	14
4.3	Проверка файла сам работы . . . . .	14
4.4	Код программы . . . . .	15
4.5	Код программы продолжение . . . . .	16
4.6	Проверка работы файла 3 . . . . .	17
4.7	Проверка работы файла 4 . . . . .	17

## **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 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов: • условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия. • безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

Флаг – это бит, принимающий значение 1 («флаг установлен»), если выполнено некоторое условие, и значение 0 («флаг сброшен») в противном случае. Флаги работают независимо друг от друга, и лишь для удобства они помещены в единый регистр — регистр флагов, отражающий текущее состояние процессора.

Инструкция `cmp` является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция `cmp` является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания: `cmp ,`

Команда условного перехода имеет вид `j label` Мнемоника перехода связана со значением анализируемых флагов или со способом формирования этих флагов.

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

### 3 Выполнение лабораторной работы

Создаю каталог для программ лабораторной работы No 7, перехожу в него и создаю файл lab7-1.asm (рис. 3.1).

```
mvgracheva@dk4n62 ~ $ mkdir ~/work/arch-pc/lab07
mvgracheva@dk4n62 ~ $ cd ~/work/arch-pc/lab07
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ touch lab7-1.asm
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $
```

Рис. 3.1: Создание каталога и файла lab7-1.asm

Ввожу листинг программы (рис. 3.2).

```
lab7-1.asm      [-M--] 41 L:[ 1+19 20/ 20] *(643 / 643b) <EOF>
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.2: Листинг программы

Запускаю файл (рис. 3.3).

```

mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 3
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ █

```

Рис. 3.3: Запуск файла

Меняю текст программы(рис. 3.4).

```

#include "in_out.asm" ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:
call quit ; вызов подпрограммы завершения █

```

Рис. 3.4: Изменение текста программы

Запускаю файл(рис. 3.5).

```

mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 1
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ █

```

Рис. 3.5: Запуск файла 2



Меняю текст программы, чтобы выходило 3, 2, 1(рис. 3.6).

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.6: Новый листинг программы

Запускаю файл(рис. 3.7).

```
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $
```

Рис. 3.7: Запуск файла 3

Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. (рис. 3.8).

```
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ touch lab7-2.asm
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $
```

Рис. 3.8: Создание каталога и файла lab7-2.asm

Листинг программы 2(рис. 3.9).

```
lab7-2.asm      [-M--] 12 L:[ 1+13 14/ 49] *(282 /1743)
#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:
; ----- 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'
```

Рис. 3.9: Листинг программы 2

Проверяю работу файла (рис. 3.10), (рис. 3.11).

```

mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 3
Наибольшее число: 50
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ █

```

Рис. 3.10: Проверка работы файла

```

mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 30
Наибольшее число: 50
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 60
Наибольшее число: 60
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ █

```

Рис. 3.11: Проверка работы файла 2

Создаю файл листинга для программы из файла lab7-2.asm. Открываю файл листинга lab7-2.lst с помощью mcedit: (рис. 3.12).

```

mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ mcedit lab7-2.lst █

```

Рис. 3.12: Команда nasm с ключом -l и mcedit

Открываем файл (рис. 3.13).

```

1          %include 'in_out.asm'
1          <1> ;----- slen -----
2          <1> ; Функция вычисления длины сообщения
3          <1> slen:
4 00000000 53          <1> push    ebx
5 00000001 89C3        <1> mov     ebx, eax
6          <1>
7          <1> nextchar:
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
27 00000013 E8E8FFFF   <1> call    slen

```

Рис. 3.13: Открытый файл lst

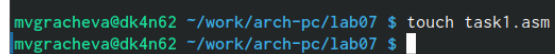
Первая строка это использование другого файла, чтобы не расписывать каждый раз программы

11 строка это переход на метку nextchar

22 строка для того, чтобы вывести сообщение

## 4 Выполнение самостоятельной работы

Создаю файл (рис. 4.1), ввожу текст (рис. 4.2), проверяю работу файла (рис. 4.3).

A terminal window with a dark background. The first line shows the command 'touch task1.asm' being executed. The second line shows the prompt after the command has finished.

```
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $ touch task1.asm
mvgracheva@dk4n62 ~/work/arch-pc/lab07 $
```

Рис. 4.1: Создание файла для сам работы

```

task1.asm [----] 17 L: [ 1+34 35/ 35] *(1109/110
#include 'in_out.asm'
section .data
msg2 db "Наименьшее число: ",0h
A dd 44
B dd 74
C dd 17
section .bss
min resb 10

section .text
global _start
_start:
; ----- Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
j1 check_B ; если 'A<C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число
check_B:
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
j1 fin ; если 'min(A,C)<B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наименьшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход

```

Рис. 4.2: Введение текста в сам работу

```

mvgracheva@dk8n69 ~/work/arch-pc/lab07 $ nasm -f elf task1.asm
mvgracheva@dk8n69 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o task1 task1.o
mvgracheva@dk8n69 ~/work/arch-pc/lab07 $ ./task1
Наименьшее число: 17

```

Рис. 4.3: Проверка файла сам работы

## Задание 2

Код программы (рис. 4.4), (рис. 4.5).

```

%include 'in_out.asm'
section .data
msg1 db 'Введите число для значения a: ',0h
msg2 db 'Введите число для значения x: ',0h

section .bss
A resb 10
X resb 10

section .text
global _start
_start:
; введите a
mov eax,msg1
call sprint
mov ecx,A
mov edx,10
call sread

mov eax,A
call atoi
mov [A],eax
; введите x
mov eax, msg2
call sprint

mov ecx,X
mov edx,10
call sread

mov eax,X
call atoi
mov [X],eax

mov ebx, 4
cmp [X],ebx
jge check

```

Рис. 4.4: Код программы

```
mov eax,[A]
mov ebx,8
add eax, ebx
call iprintLF
call quit
```

**check:**

```
mov eax,[A]
mov ebx,[X]
mul ebx
call iprintLF
```

```
call quit
```

Рис. 4.5: Код программы продолжение

Проверка работы файла (рис. 4.6), (рис. 4.7).



```
mvgracheva@dk8n51 ~/work/arch-pc/lab07 $ nasm -f elf smth.asm
mvgracheva@dk8n51 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o smth smth.o
mvgracheva@dk8n51 ~/work/arch-pc/lab07 $ ./smth
Введите число для значения а: 1
Введите число для значения х: 1
5
```

Рис. 4.6: Проверка работы файла 3

```
mvgracheva@dk8n51 ~/work/arch-pc/lab07 $ ./smth
Введите число для значения а: 1
Введите число для значения х: 7
7
mvgracheva@dk8n51 ~/work/arch-pc/lab07 $
```

Рис. 4.7: Проверка работы файла 4

## 5 Выводы

Изучила команды условного и безусловного переходов. Приобрела навыков написания программ с использованием переходов. Ознакомилась с назначением и структурой файла листинга.

## **Список литературы**