

# **Лабораторная работа №7**

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

Бунин Арсений Викторович

# Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Самостоятельная работа	13
6	Выводы	19
	Список литературы	20

## Список иллюстраций

4.1	Создание файла программы . . . . .	8
4.2	Код программы . . . . .	9
4.3	Пример программы с использованием jmp . . . . .	9
4.4	Измененная программа . . . . .	10
4.5	Результат работы измененной программы . . . . .	10
4.6	Код программы . . . . .	11
4.7	Работа программы . . . . .	11
4.8	Работа программы . . . . .	12
4.9	Работа программы . . . . .	12
5.1	Код программы . . . . .	14
5.2	Код программы.Продолжение . . . . .	15
5.3	Работа программы . . . . .	16
5.4	Код программы . . . . .	17
5.5	Работа программы . . . . .	18

## **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 Задание

1. Создать файл на языке Ассемблер, выводящий текст с использованием `jmp`
2. Создать файл на языке Ассемблер, сравнивающий числа с использованием `jmp`
3. Выполнить индивидуальное задание по написанию программы на Ассемблере
4. Загрузить файлы на github

### 3 Теоретическое введение

Безусловный переход выполняется инструкцией `jmp`, которая включает в себя адрес перехода, куда следует передать управление для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов. 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. 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

## 4 Выполнение лабораторной работы

Создаем исполняемый файл(рис. 4.1)

```
[arsenii@fedora arch-pc]$ mkdir labs/lab07  
[arsenii@fedora arch-pc]$ cd labs/lab07  
[arsenii@fedora lab07]$ touch lab7-1.asm  
[arsenii@fedora lab07]$
```

Рис. 4.1: Создание файла программы

Код программы, выводящей строки в определенной последовательности(рис. 4.2)



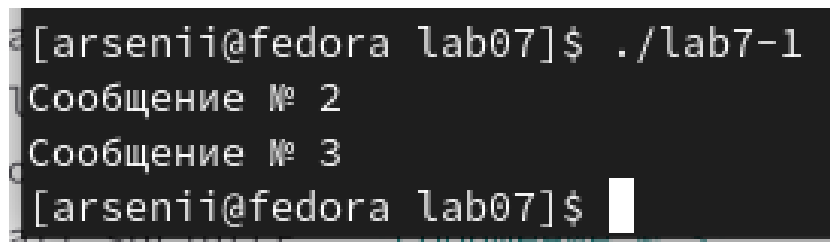
```

%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 sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 4.2: Код программы

Результат работы программы(рис. 4.3)



```

[arsenii@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[arsenii@fedora lab07]$

```

Рис. 4.3: Пример программы с использованием jmp

Измененный текст программы (рис. 4.4).

```

%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 _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. 4.4: Измененная программа

Результат работы программы, выводящий строки в другой последовательности(рис. 4.5).

```

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

```

Рис. 4.5: Результат работы измененной программы

Программа, выводящая строки в обратной последовательности (рис. 4.6)

```
%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 _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. 4.6: Код программы

Результат работы программы(рис. 4.7)

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

Рис. 4.7: Работа программы

Результат работы программы, сравнивающей числа(рис. 4.8)

```
[arsenii@fedora lab07]$ nasm -f elf lab7-2.asm
[arsenii@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[arsenii@fedora lab07]$ ./lab7-2
Введите В: 100
Наибольшее число: 100
[arsenii@fedora lab07]$ ./lab7-2
Введите В: 35
Наибольшее число: 50
[arsenii@fedora lab07]$
```

Рис. 4.8: Работа программы

При выполнении данной команды (рис. 4.9) создается файл с расширением `lst`. В файл добавляются все встроенные библиотеки, номера строк, адрес и машинный код каждой команды

```
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
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
13         <1> finished:
14         0000000B 29D8       <1>      sub     eax, ebx
15         0000000D 5B          <1>      pop     ebx
16         0000000E C3          <1>      ret
17
18
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 E8E8FFFFFF <1>      call    slen
28
29         00000018 89C2       <1>      mov     edx, eax
30         0000001A 58          <1>      pop     eax
31
32         0000001B 89C1       <1>      mov     ecx, eax
33         0000001D BB01000000 <1>      mov     ebx, 1
```

Рис. 4.9: Работа программы

## 5 Самостоятельная работа

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных  $a, b$  и  $c$ . Значения переменных выбрать из в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу.

Код программы(рис. 5.1 и рис. 5.2)

```

1  %include 'in_out.asm'
2  section .data
3  msg1 db 'Введите A: ',0h
4  msg2 db 'Введите B: ',0h
5  msg3 db 'Введите C: ',0h
6  msgres db 'Наименьшее число: ',0h
7  section .bss
8  min resb 10
9  A resb 10
10 B resb 10
11 C resb 10
12 section .text
13 global _start
14 _start:
15 ; ----- Вывод сообщения 'Введите A: '
16 mov eax,msg1
17 call sprint
18 ; ----- Ввод 'A'
19 mov ecx,A
20 mov edx,10
21 call sread
22 mov eax,A
23 call atoi
24 mov [A],eax ; запись числа в 'A'
25 ; ----- Ввод 'B'
26 mov eax,msg2
27 call sprint
28 mov ecx,B
29 mov edx,10
30 call sread
31 mov eax,B
32 call atoi
33 mov [B],eax ; запись числа в 'B'

```

Рис. 5.1: Код программы

```

34 ; ----- Ввод 'C'
35 mov eax,msg3
36 call sprint
37 mov ecx,C
38 mov edx,10
39 call sread
40 mov eax,C
41 call atoi
42 mov [C],eax ; запись числа в 'C'
43 ; ----- Записываем 'A' в переменную 'min'
44 mov ecx,[A] ; 'ecx = A'
45 mov [min],ecx ; 'min = A'
46 ; ----- Сравниваем 'A' и 'C'
47 cmp ecx,[C] ; Сравниваем 'A' и 'C'
48 jl check_B ; если 'A<C', то переход на метку 'check_B',
49 mov ecx,[C] ; иначе 'ecx = C'
50 mov [min],ecx ; 'min = C'
51 check_B:
52 mov ecx,[min]
53 cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
54 jl fin ; если 'min(A,C)<B', то переход на 'fin',
55 mov ecx,[B] ; иначе 'ecx = B'
56 mov [min],ecx
57 ; ----- Вывод результата
58 fin:
59 mov eax, msgres
60 call sprint ; Вывод сообщения 'Наименьшее число: '
61 mov eax,[min]
62 call iprintLF ; Вывод 'min(A,B,C)'
63 call quit ; Выход

```

Рис. 5.2: Код программы.Продолжение

Результат работы программы (рис. 5.3)

```
[arsenii@fedora lab07]$ nasm -f elf lab7-3.asm
[arsenii@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[arsenii@fedora lab07]$ ./lab7-3
Введите A: 46
Введите B: 32
Введите C: 74
Наименьшее число: 32
[arsenii@fedora lab07]$
```

Рис. 5.3: Работа программы

2. 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

Код программы(рис. 5.4)



```

1  %include 'in-out.asm'
2  section .data
3  msg1 db 'Введите x: ',0h
4  msg2 db 'Введите a: ',0h
5  msgres db 'Результат: ',0h
6  section .bss
7  res resb 10
8  X resb 10
9  A resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите X: '
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод 'A'
17 mov ecx,X
18 mov edx,10
19 call sread
20 mov eax,X
21 call atoi
22 mov [X],eax ; запись числа в 'X'
23 ; ----- Ввод 'A'
24 mov eax,msg2
25 call sprint
26 mov ecx,A
27 mov edx,10
28 call sread
29 mov eax,A
30 call atoi
31 mov [A],eax ; запись числа в 'A'
32 ; ----- Записываем 'A' в переменную 'edi'
33 mov edi,[X] ; 'ecx = X'
34 ; ----- Сравниваем 'A' и 'X'
35 cmp edi,[A]; Сравниваем 'A' и 'C'
36 jbe fin ; если 'X<=A', то переход на метку 'fin',
37 ; положить в res значение x и a
38 add edi,[A]
39 ; ----- Вывод результата
40 fin:
41 mov eax, msgres
42 call sprint ; Вывод сообщения
43 mov eax,edi
44 call iprintLF
45 call quit ; Выход

```

Рис. 5.4: Код программы

Результат работы программы (рис. 5.5)

```
наибольшее значение 11
[arsenii@fedora lab07]$ ./lab7-4
Введите x: 4
Введите a: 5
Результат: 4
[arsenii@fedora lab07]$ ./lab7-4
Введите x: 3
Введите a: 2
Результат: 5
[arsenii@fedora lab07]$
```

Рис. 5.5: Работа программы

## 6 Выводы

Изучили команды условного и безусловного переходов. Приобрели навыки написания программ с использованием переходов. Ознакомились с назначением и структурой файла листинга.

## **Список литературы**