

Отчёт по лабораторной работе №7

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

Мантуров Татархан Бесланович

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выполнение самостоятельной работы	13
5	Выводы	16

Список иллюстраций

3.1	Создание файла	7
3.2	Файл листинга 7.1.	8
3.3	Создание исполняемого файла	8
3.4	Текст листинга 7.2.	9
3.5	Создание исполняемого файла	9
3.6	Измененный текст листинга	10
3.7	Создание исполняемого файла	10
3.8	Текст листинга 7.3.	10
3.9	Создание исполняемого файла	11
3.10	Создание файла листинга	11
3.11	Файл листинга	11
3.12	Ошибка компиляции файла листинга	12
4.1	Задание №1	13
4.2	Задание №2	15

Список таблиц

1 Цель работы

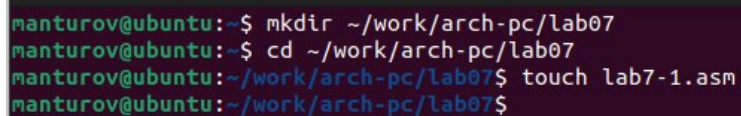
Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Задание

1. 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 № 7. Создайте исполняемый файл и проверьте его работу.
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.

3 Выполнение лабораторной работы

1. Создадим каталог для программ лабораторной работы № 7, перейдем в него и создадим файл lab7-1.asm:(3.1)



```
manturov@ubuntu:~$ mkdir ~/work/arch-pc/lab07
manturov@ubuntu:~$ cd ~/work/arch-pc/lab07
manturov@ubuntu:~/work/arch-pc/lab07$ touch lab7-1.asm
manturov@ubuntu:~/work/arch-pc/lab07$
```

Рис. 3.1: Создание файла

2. Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp. Введем в файл lab7-1.asm текст программы из листинга 7.1.(3.2)

```

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

```

Рис. 3.2: Файл листинга 7.1.

3. Создадим исполняемый файл и запустим его. Результат работы данной программы будет следующим:(3.3)

```

manturov@ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3

```

Рис. 3.3: Создание исполняемого файла

Таким образом, использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения. Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала 'Сообщение № 2', потом 'Сообщение № 1' и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения

№ 1) и после вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`). Изменим текст программы в соответствии с листингом 7.2.(3.4)

```
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'
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'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.4: Текст листинга 7.2.

4. Создадим исполняемый файл и проверим его работу.(3.5)

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

Рис. 3.5: Создание исполняемого файла

Изменим текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим: Сообщение № 3 Сообщение № 2 Сообщение № 1(3.6)

```

%include 'in_out.asm' ; подключение внешнего
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 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'

```

Рис. 3.6: Измененный текст листинга

5. Создадим исполняемый файл и проверим его работу.(3.7)

```

manturov@ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 3.7: Создание исполняемого файла

6. Создадим файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучим текст программы из листинга 7.3 и введем в lab7-2.asm.(3.8)

```

manturov@ubuntu:~/work/arch-pc/lab07$ touch lab7-2.asm
manturov@ubuntu:~/work/arch-pc/lab07$ mc

```

Рис. 3.8: Текст листинга 7.3.

Создадим исполняемый файл и проверим его работу для разных значений В.(3.9)

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

Рис. 3.9: Создание исполняемого файла

7. Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в командной строке. Создадим файл листинга для программы из файла lab7-2.asm(3.10)

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

Рис. 3.10: Создание файла листинга

Откроем файл листинга lab7-2.lst с помощью любого текстового редактора, например mcedit:(3.11)

```
1      %include 'in_out.asm'
2
3      <1> ;----- slen -----
4      <1> ; Функция вычисления длины сообщения
5      <1> slen:
6      5 00000000 53      <1> push    ebx
7      6 00000001 89C3    <1> mov     ebx, eax
8      <1>
9      9 00000003 803800  <1> nextchar:
10     10 00000006 7403    <1> cmp     byte [eax], 0
11     11 00000008 40      <1> jz      finished
12     12 00000009 EBF8    <1> inc     eax
13     13              <1> jmp     nextchar
14     14              <1>
15     15 0000000B 29D8    <1> finished:
16     16 0000000D 5B      <1> sub     eax, ebx
17     17 0000000E C3      <1> pop     ebx
18     18              <1> ret
19     19              <1>
20     20              <1> ;----- sprint -----
21     21              <1> ; Функция печати сообщения
22     22              <1> ; входные данные: mov eax, <message>
```

Рис. 3.11: Файл листинга

Удалим в файле листинга один операнд и проверим:(3.12)

```
manturov@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm  
lab7-2.asm:34: error: invalid combination of opcode and operands
```

Рис. 3.12: Ошибка компиляции файла листинга

4 Выполнение самостоятельной работы

(Вариант №20) 1. Напишем программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выберем из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создадим исполняемый файл и проверим его работу.(4.1)

```
manturov@ubuntu:~/work/arch-pc/lab07$ ./lab7-3
Enter number A: 95
Enter number B: 2
Enter number C: 61
The smallest number is: 2
```

```
%include 'in_out.asm'

section .data
    msg db 'Enter number A: ', 0
    msg2 db 'Enter number B: ', 0
    msg3 db 'Enter number C: ', 0
    result_msg db 'The smallest number is: ', 0

section .bss
    A resb 10
    B resb 10
    C resb 10
    min resb 10

section .text
    global _start

_start:
    ; Input for A
    mov eax, msg
```

Рис. 4.1: Задание №1

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.(4.2)

```
Result: 5
manturov@ubuntu:~/work/arch-pc/lab07$ ./laba
Введите X: 2
Введите A: 1
Result: 1
manturov@ubuntu:~/work/arch-pc/lab07$ 
manturov@ubuntu:~/work/arch-pc/lab07$ ./laba
Введите X: 1
Введите A: 2
Result: 5
```

```

%include 'in_out.asm'

section .data
msg db 'Result: ', 0
msg1 db 'Введите X: ', 0
msg2 db 'Введите A: ', 0
a dd 79
b dd 83
c dd 41

section .bss
x resb 10
result resb 10

section .text
global _start

_start:
    ; Input for x
    mov eax, msg1

```

Рис. 4.2: Задание №2

5 Выводы

Я изучил команды условного и безусловного переходов. Приобрел навыки написания программ с использованием переходов. Ознакомился с назначением и структурой файла листинга.