

Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютера

Булыгин Николай Александрович

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Реализация переходов в NASM	7
3.2	Изучение структуры файлы листинга	13
3.3	Выполнение самостоятельной работы	15
4	Выводы	21

Список иллюстраций

3.1	lab7-1.asm	7
3.2	Вывод программы	8
3.3	Новый вывод программы	10
3.4	Порядок вывода 3 2 1	11
3.5	lab7-2.asm	11
3.6	lab7-2	13
3.7	Создание файла листинга	13
3.8	lab7-2.lst	14
3.9	Строка 16	14
3.10	Строка 21	14
3.11	Строка 30	14
3.12	Ошибка	15
3.13	Сообщение об ошибке	15
3.14	Задание 1	15
3.15	Задание 2	18

Список таблиц

1 Цель работы

Целью работы является изучение команд условного и безусловного переходов, приобретение навыков написания программ с использованием переходов и знакомство с назначением и структурой файла листинга.

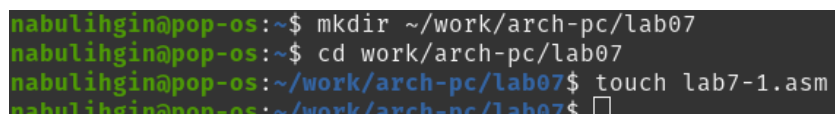
2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлы листинга
3. Выполнение самостоятельной работы

3 Выполнение лабораторной работы

3.1 Реализация переходов в NASM

Создаю каталог для программ в этой лабораторной работе, перехожу в него и создаю файл lab7-1.asm (рис. 3.1).



```
nabulihgin@pop-os:~$ mkdir ~/work/arch-pc/lab07
nabulihgin@pop-os:~$ cd work/arch-pc/lab07
nabulihgin@pop-os:~/work/arch-pc/lab07$ touch lab7-1.asm
nabulihgin@pop-os:~/work/arch-pc/lab07$
```

Рис. 3.1: lab7-1.asm

Ввожу в файл lab7-1.asm текст программы:

```
%include 'in_out.asm'

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
    call sprintf

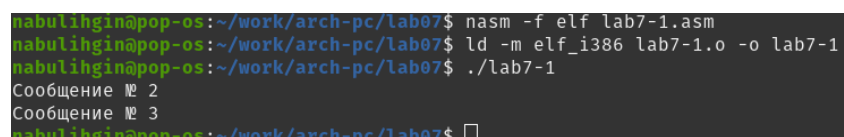
_label2:
    mov eax, msg2
    call sprintf

_label3:
    mov eax, msg3
    call sprintf

_end:
    call quit

```

Создаю исполняемый файл и запускаю его. Выводится сначала Сообщение 2, потом Сообщение 3, так как `jmp _label2` осуществляет переход к `_label2`, что пропускает `_label1` (рис. 3.2).



```

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

```

Рис. 3.2: Вывод программы

Меняю текст `lab7-1.asm` в соответствие с листингом 7.2:

```

#include 'in_out.asm'

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
```

```
    call sprintf
```

```
    jmp _end
```

```
_label2:
```

```
    mov eax, msg2
```

```
    call sprintf
```

```
    jmp _label1
```

```
_label3:
```

```
    mov eax, msg3
```

```
    call sprintf
```

```
_end:
```

```
    call quit
```

Создаю исполняемый файл и запускаю его. Выводится сначала Сообщение 2, потом Сообщение 1, так как `jmp _label2` осуществляет переход к `_label2`, затем `jmp _label1` переходит на `label1` и `jmp end` вызывает завершение (рис. 3.3).

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

Рис. 3.3: Новый вывод программы

Снова меняю текст программы, чтобы сообщения выводились в порядке 3 2 1:

```
%include 'in_out.asm'
```

```
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
```

```
    call sprintfLF
```

```
    jmp _end
```

```
_label2:
```

```
    mov eax, msg2
```

```
    call sprintfLF
```

```
    jmp _label1
```

```
_label3:
```

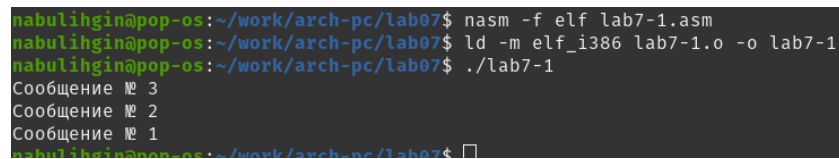
```

mov eax, msg3
call sprintf
jmp _label2

_end:
call quit

```

Снова создаю исполняемый файл и запускаю его, вывод соответствует условию задания (рис. 3.4).



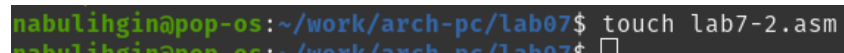
```

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

```

Рис. 3.4: Порядок вывода 3 2 1

Создаю файл lab7-2.asm (рис. 3.5).



```

nabulihgin@pop-os:~/work/arch-pc/lab07$ touch lab7-2.asm
nabulihgin@pop-os:~/work/arch-pc/lab07$ █

```

Рис. 3.5: lab7-2.asm

Ввожу в него текст программы из листинга 7.3:

```

#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:
    mov eax,msg1
    call sprint

    mov ecx,B
    mov edx,10
    call sread

    mov eax,B
    call atoi
    mov [B],eax

    mov ecx,[A]
    mov [max],ecx

    cmp ecx,[C]
    jg check_B
    mov ecx,[C]
    mov [max],ecx

check_B:
    mov eax,max
    call atoi
    mov [max],eax

    mov ecx,[max]
    cmp ecx,[B]
    jg fin

```

```

    mov ecx, [B]
    mov [max], ecx

fin:
    mov eax, msg2
    call sprint
    mov eax, [max]
    call iprintLF
    call quit

```

Создаю исполняемый файл и проверяю его работу для разных значений В (рис. 3.6).

```

nabulihgin@pop-os:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
nabulihgin@pop-os:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
nabulihgin@pop-os:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 63
Наибольшее число: 63
nabulihgin@pop-os:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 66
Наибольшее число: 66
nabulihgin@pop-os:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 20
Наибольшее число: 50
nabulihgin@pop-os:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 51
Наибольшее число: 51
nabulihgin@pop-os:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 50
Наибольшее число: 50
nabulihgin@pop-os:~/work/arch-pc/lab07$ █

```

Рис. 3.6: lab7-2

3.2 Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.asm (рис. 3.7).

```

nabulihgin@pop-os:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
nabulihgin@pop-os:~/work/arch-pc/lab07$ █

```

Рис. 3.7: Создание файла листинга

Открываю полученный файл листинга lab7-2.lst в mcedit (рис. 3.8).

```

1 1 %include 'in_out.asm'
2 2 <1> ;----- slen -----
3 3 <1> ; Функция вычисления длины сообщения
4 4 <1> slen:
5 5 00000000 53 <1> push ebx
6 6 00000001 89C3 <1> mov ebx, eax
7 7 <1>
8 8 <1> nextchar:
9 9 00000003 803800 <1> cmp byte [eax], 0
10 10 00000006 7403 <1> jz finished
11 11 00000008 40 <1> inc eax
12 12 00000009 EBF8 <1> jmp nextchar
13 13 <1>
14 14 <1> finished:
15 15 0000000B 29D8 <1> sub eax, ebx
16 16 0000000D 5B <1> pop ebx
17 17 0000000E C3 <1> ret
18 18 <1>
19 19 <1>
20 20 <1> ;----- sprint -----
21 21 <1> ; Функция печати сообщения
22 22 <1> ; входные данные: mov eax,<message>
23 23 <1> sprint:
24 24 0000000F 52 <1> push edx
25 25 00000010 51 <1> push ecx
26 26 00000011 53 <1> push ebx
27 27 00000012 50 <1> push eax
28 28 00000013 E8E8FFFFFF <1> call slen

```

Рис. 3.8: lab7-2.lst

Объясняю содержимое трех строк файла по выбору:

Строка на 16 месте, адрес - 000000F2, машинный код - B9[0A000000], mov ecx,B - исходный текст программы, в регистр ecx вносится значение переменной B (рис. 3.9).

```

16 000000F2 B9[0A000000] mov ecx,B

```

Рис. 3.9: Строка 16

Строка на 21 месте, адрес - 00000106, машинный код - E891FFFFFF, call atoi - исходный текст программы, символ в строке выше переводится в число (рис. 3.10).

```

21 00000106 E891FFFFFF call atoi

```

Рис. 3.10: Строка 21

Строка на 30 месте, адрес - 0000012A, машинный код - 890D[00000000], mov [max],ecx - исходный текст программы, в регистр ecx вносится число, хранившееся в переменной max (рис. 3.11).

```

30 0000012A 890D[00000000] mov [max],ecx

```

Рис. 3.11: Строка 30

В строке 44 удаляю операнд msg2, выполняю трансляцию с получением файла листинга и получаю ошибку (рис. 3.12).

```
nabulihgin@pop-os:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:44: error: invalid combination of opcode and operands
nabulihgin@pop-os:~/work/arch-pc/lab07$
```

Рис. 3.12: Ошибка

В файле листинга добавляется сообщение об ошибке (рис. 3.13).

```
44                                     mov eax
44      *****                      error: invalid combination of opcode and operands
45 00000159 E8B1FEFFFF                call sprint
```

Рис. 3.13: Сообщение об ошибке

3.3 Выполнение самостоятельной работы

Создаю файл task1.asm и пишу программу для нахождения наименьшего из 3 чисел. У меня второй вариант, поэтому в программе задаю числа A = 82, B = 59, C = 61. Программа выводит 59 (рис. 3.14).

```
nabulihgin@pop-os:~/work/arch-pc/lab07$ nasm -f elf task1.asm
nabulihgin@pop-os:~/work/arch-pc/lab07$ ld -m elf_i386 task1.o -o task1
nabulihgin@pop-os:~/work/arch-pc/lab07$ ./task1
A: 82
B: 59
C: 61
Smallest num: 59
nabulihgin@pop-os:~/work/arch-pc/lab07$
```

Рис. 3.14: Задание 1

Код программы:

```
%include 'in_out.asm'

section .data

msga db 'A: ',0h
msgb db 'B: ',0h
msgc db 'C: ',0h
msg1 db "Smallest num: ",0h
```

```

section .bss
min resb 10
A resb 10
B resb 10
C resb 10
section .text
global _start
_start:

mov eax,msga
call sprint
mov ecx,A
mov edx, 10
call sread
mov eax,msgb
call sprint
mov ecx,B
call sread
mov eax,msgc
call sprint
mov ecx,C
call sread

mov eax,A
call atoi
mov [A],eax
mov eax,B
call atoi
mov [B],eax

```



```

mov eax,C
call atoi
mov [C],eax

mov ecx,[A]
add ecx,[B]
add ecx,[C]
mov [min],ecx

jmp cmp_a_b

min_below_c:
jmp fin

c_below_min:
mov eax,[C]
mov [min],eax
jmp fin

a_below_b:
mov eax,[A]
mov [min],eax
jmp cmp_min_c

b_below_a:
mov eax,[B]
mov [min],eax
jmp cmp_min_c

```

```

cmp_min_c:
mov eax, [C]
cmp [min], eax
jb min_below_c
jg c_below_min

```

```

cmp_a_b:
mov eax, [B]
cmp [A], eax
jb a_below_b
jg b_below_a

```

```

fin:
mov eax, msg1
call sprint
mov eax, [min]
call iprintLF
call quit

```

Создаю файл task2.asm и пишу программу для вычисления значения функции второго варианта. При значениях $x = 5$, $a = 7$ она выводит $a - 1 = 6$, при значениях $x = 6$, $a = 4$ она выводит $x - 1 = 5$, следовательно, она работает корректно (рис. 3.15).

```

nabulihgin@pop-os:~/work/arch-pc/lab07$ nasm -f elf task2.asm
nabulihgin@pop-os:~/work/arch-pc/lab07$ ld -m elf_i386 -o task2 task2.o
nabulihgin@pop-os:~/work/arch-pc/lab07$ ./task2
Enter x: 5
Enter a: 7
Result: 6
nabulihgin@pop-os:~/work/arch-pc/lab07$ ./task2
Enter x: 6
Enter a: 4
Result: 5
nabulihgin@pop-os:~/work/arch-pc/lab07$ 

```

Рис. 3.15: Задание 2

Код программы:

```

#include 'in_out.asm'

section .data
msgx db 'Enter x: ',0h
msga db 'Enter a: ',0h
msg1 db "Result: ",0h

section .bss
x resb 10
a resb 10
res resb 10

section .text
global _start
_start:
mov eax,msgx
call sprint
mov ecx,x
mov edx,10
call sread
mov eax,msga
call sprint
mov ecx,a
call sread
mov eax,x
call atoi
mov [x],eax
mov eax,a
call atoi
mov [a],eax

```

```
mov eax, [a]
mov ebx, [x]
cmp ebx, eax
jb x_below_a
jae x_above_a
```

```
x_below_a:
mov ecx, [a]
mov [res], ecx
mov ecx, 1
sub [res], ecx
jmp fin
```

```
x_above_a:
mov ecx, [x]
mov [res], ecx
mov ecx, 1
sub [res], ecx
jmp fin
```

```
fin:
mov eax, msg1
call sprint
mov eax, [res]
call iprintLF
call quit
```

4 Выводы

Я изучил команды условного и безусловного переходов, приобрел навыки написания программ с использованием переходов и познакомился с назначением и структурой файла листинга.