

Отчёт по лабораторной работе 7

Команды безусловного и условного переходов в Nasm.

Калашникова Дарья Викторовна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Условные переходы	11
2.3	Изучение структуры файла листинга	13
2.4	Самостоятельное задание	16
3	Выводы	21

Список иллюстраций

2.1	Создан каталог	6
2.2	Программа lab7-1.asm	7
2.3	Запуск программы lab7-1.asm	7
2.4	Программа lab7-1.asm	8
2.5	Запуск программы lab7-1.asm	9
2.6	Программа lab7-1.asm	10
2.7	Запуск программы lab7-1.asm	10
2.8	Программа lab7-2.asm	12
2.9	Запуск программы lab7-2.asm	13
2.10	Файл листинга lab7-2	14
2.11	Ошибка трансляции lab7-2	15
2.12	Файл листинга с ошибкой lab7-2	16
2.13	Программа task1.asm	17
2.14	Запуск программы task1.asm	17
2.15	Программа task2.asm	19
2.16	Запуск программы task2.asm	20

Список таблиц

1 Цель работы

Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

2.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы № 7 и файл lab7-1.asm.
(рис. 2.1)



Рис. 2.1: Создан каталог

Инструкция `jmp` в NASM используется для реализации безусловных переходов. Пример программы, демонстрирующей эту инструкцию, приведен в файле lab7-1.asm. (рис. 2.2)

```

lab7-1.asm      [----]  9 L:[  1+24  25/ 25] *(329 / 329b) <E
#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 sprintLF

_label2:
mov eax, msg2
call sprintLF

_label3:
mov eax, msg3
call sprintLF

_end:
call quit

```

Рис. 2.2: Программа lab7-1.asm

Создаю исполняемый файл и запускаю его. (рис. 2.3)

```

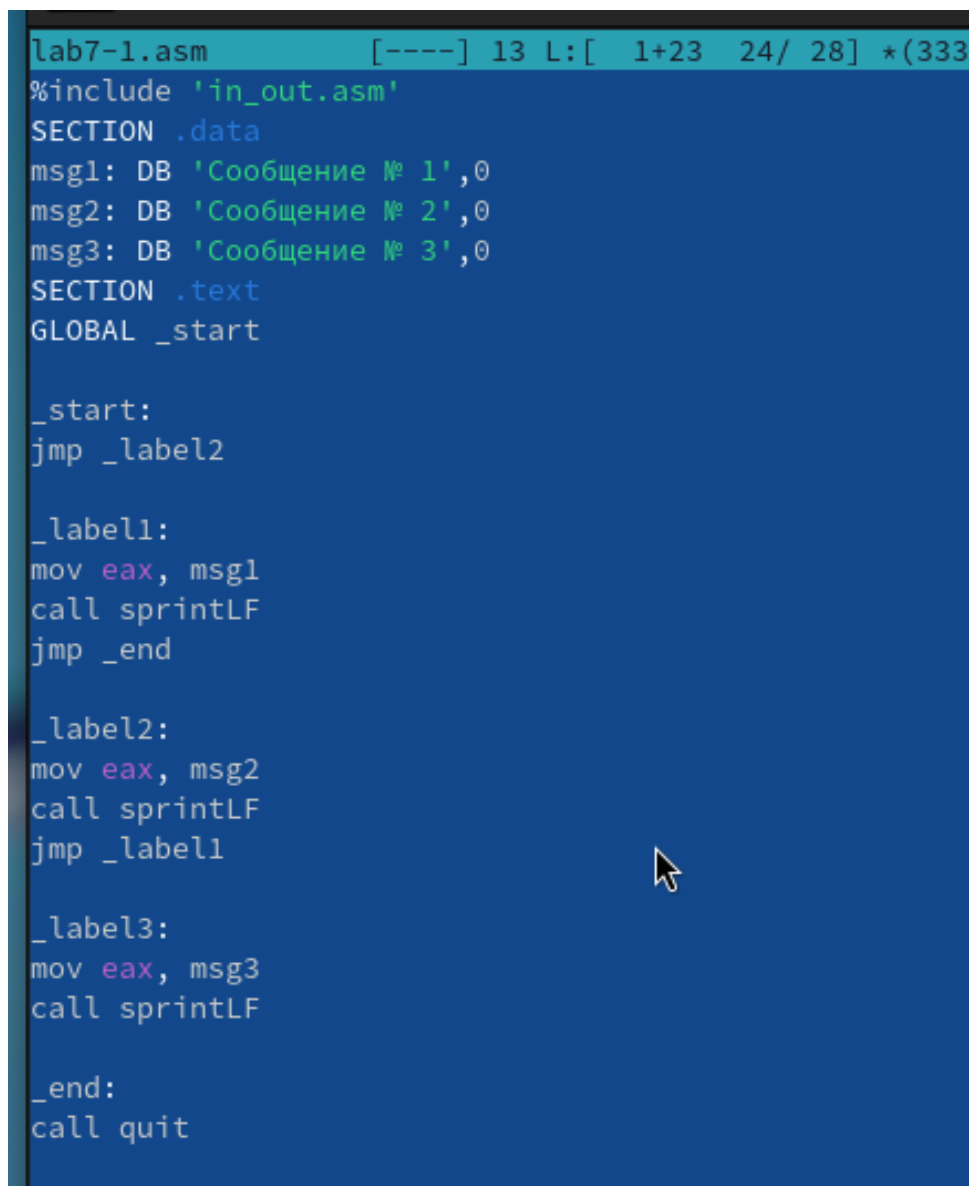
d_kalashnikova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
d_kalashnikova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
d_kalashnikova@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
d_kalashnikova@fedora:~/work/arch-pc/lab07$

```

Рис. 2.3: Запуск программы lab7-1.asm

Инструкция `jmp` позволяет осуществлять переходы как вперед, так и назад. Для изменения последовательности вывода программы добавляю метки `_label1` и `_end`. Таким образом, вывод программы изменится: сначала отобразится сообщение № 2, затем сообщение № 1, и программа завершит работу.

Обновляю текст программы согласно листингу 7.2. (рис. 2.4, рис. 2.5)



```
lab7-1.asm      [----] 13 L:[ 1+23 24/ 28] *(333
%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 sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.4: Программа lab7-1.asm


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

Рис. 2.5: Запуск программы lab7-1.asm

Дорабатываю текст программы для вывода следующих сообщений:

Сообщение № 3

Сообщение № 2

Сообщение № 1

Результат показан на рисунках (рис. 2.6, рис. 2.7).

```

lab7-1.asm [----] 11 L: [ 1+24 25/ 25
#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 sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit

```

Рис. 2.6: Программа lab7-1.asm

```

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

```

Рис. 2.7: Запуск программы lab7-1.asm

Использование инструкции `jmp` обеспечивает переходы независимо от условий. Однако для реализации условных переходов требуется использование дополнительных инструкций.

2.2 Условные переходы

Для демонстрации условных переходов создаю программу, определяющую максимальное значение среди трех переменных: А, В и С. Значения А и С задаются в программе, а В вводится с клавиатуры. Результаты работы программы представлены на рисунках (рис. 2.8, рис. 2.9).

```
lab7-2.asm [----] 9 L: [ 16+33 49/ 49] *(1056/1056b) <EOF>
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A]
mov [max],ecx
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi
mov [max],eax
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx
; ----- Вывод результата
fin:
mov eax,msg2
call sprint
mov eax,[max]
call iprintLF
call quit
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перезагрузить 7Поиск 8Удалить

Рис. 2.8: Программа lab7-2.asm

```
d_kalashnikova@fedora:~/work/arch-pc/lab07$  
d_kalashnikova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm  
d_kalashnikova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2  
d_kalashnikova@fedora:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 10  
Наибольшее число: 50  
d_kalashnikova@fedora:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 40  
Наибольшее число: 50  
d_kalashnikova@fedora:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 80  
Наибольшее число: 80  
d_kalashnikova@fedora:~/work/arch-pc/lab07$
```

Рис. 2.9: Запуск программы lab7-2.asm

2.3 Изучение структуры файла листинга

Для получения файла листинга указываю ключ `-l` при ассемблировании. Результат ассемблирования программы lab7-2.asm представлен на рисунке (рис. 2.10).

```
mc [d_kalashnikova@fedora]:~/work/arch-pc/lab07
lab7-2.lst [----] 0 L:[184+33 217/225] *(13323/13771b) 0032 0x020 [*][X]
 9 0000000A <res Ah> B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 000000E8 B8[00000000] mov eax,msg1
15 000000ED E81DFFFFFF call sprint
16 ; ----- Ввод 'B'
17 000000F2 B9[0A000000] mov ecx,B
18 000000F7 BA0A000000 mov edx,10
19 000000FC E842FFFFFF call sread
20 ; ----- Преобразование 'B' из символа в число
21 00000101 B8[0A000000] mov eax,B
22 00000106 E891FFFFFF call atoi
23 0000010B A3[0A000000] mov [B],eax
24 ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000] mov ecx,[A]
26 00000116 890D[00000000] mov [max],ecx
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000] cmp ecx,[C]
29 00000122 7F0C jg check_B
30 00000124 8B0D[39000000] mov ecx,[C]
31 0000012A 890D[00000000] mov [max],ecx
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 00000130 B8[00000000] mov eax,max
35 00000135 E862FFFFFF call atoi
36 0000013A A3[00000000] mov [max],eax
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013F 8B0D[00000000] mov ecx,[max]
39 00000145 3B0D[0A000000] cmp ecx,[B]
40 0000014B 7F0C jg fin
41 0000014D 8B0D[0A000000] mov ecx,[B]
42 00000153 890D[00000000] mov [max],ecx
```

Рис. 2.10: Файл листинга lab7-2

Анализируя структуру листинга, можно увидеть соответствие строк кода и их машинного представления. Например:

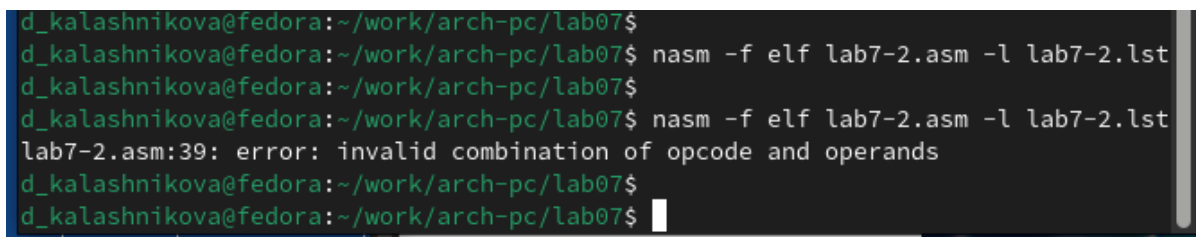
- **Строка 203:**
 - Номер строки: 28
 - Адрес: 0000011C
 - Машинный код: 3B0D[39000000]
 - Команда: `cmp ecx,[C]`
- **Строка 204:**

- Номер строки: 29
- Адрес: 00000122
- Машинный код: 7F0C
- Команда: jg check_B

• **Строка 205:**

- Номер строки: 30
- Адрес: 00000124
- Машинный код: 8B0D[39000000]
- Команда: mov ecx,[C]

Далее изменяю инструкцию с двумя операндами, удаляя один, и повторяю трансляцию. Возникает ошибка, результат которой отображен на рисунках (рис. 2.11, рис. 2.12).



```
d_kalashnikova@fedora:~/work/arch-pc/lab07$
d_kalashnikova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
d_kalashnikova@fedora:~/work/arch-pc/lab07$
d_kalashnikova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:39: error: invalid combination of opcode and operands
d_kalashnikova@fedora:~/work/arch-pc/lab07$
d_kalashnikova@fedora:~/work/arch-pc/lab07$
```

Рис. 2.11: Ошибка трансляции lab7-2

```
lab7-2.lst      [----]  0 L:[193+26 219/226] *(13465/13859b) 0032 0x020  [*][X]
18 000000F7 BA0A000000      mov edx,10
19 000000FC E842FFFFFF      call spread
20                          ; ----- Преобразование 'B' из символа в число
21 00000101 B8[0A000000]    mov eax,B
22 00000106 E891FFFFFF      call atoi
23 0000010B A3[0A000000]    mov [B],eax
24                          ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000]  mov ecx,[A].
26 00000116 890D[00000000]  mov [max],ecx
27                          ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000]  cmp ecx,[C]
29 00000122 7F0C            jg check_B
30 00000124 8B0D[39000000]  mov ecx,[C].
31 0000012A 890D[00000000]  mov [max],ecx
32                          ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 00000130 B8[00000000]    mov eax,max
35 00000135 E862FFFFFF      call atoi
36 0000013A A3[00000000]    mov [max],eax
37                          ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013F 8B0D[00000000]  mov ecx,[max]
39                          cmp ecx,
39 *****
39                          error: invalid combination of opcode and operands
40 00000145 7F0C            jg fin
41 00000147 8B0D[0A000000]  mov ecx,[B]
42 0000014D 890D[00000000]  mov [max],ecx
43                          ; ----- Вывод результата
44 fin:
45 00000153 B8[13000000]    mov eax,msg2
46 00000158 E8B2FFFFFF      call sprint
47 0000015D A1[00000000]    mov eax,[max]
48 00000162 E81FFFFFFF      call iprintLF
49 00000167 E86FFFFFFF      call quit
```

Рис. 2.12: Файл листинга с ошибкой lab7-2

2.4 Самостоятельное задание

1. Напишите программу, которая находит наименьшее значение из трех переменных a, b и c для следующих значений:

Вариант 9: 24,98,15.

Результат работы программы показан на рисунках (рис. 2.13, рис. 2.14).


```

task1.asm [-----] 18 L: [ 37+24 61/ 70] *(879 / 951b[*] [X]
    mov eax,msgC
    call sprint
    mov ecx,C
    mov edx,80
    call sread.
    mov eax,C
    call atoi
    mov [C],eax...
....
    mov ecx,[A]
    mov [min],ecx

    cmp ecx, [B]
    jl check_C
    mov ecx, [B]
    mov [min], ecx

check_C:
    cmp ecx, [C]
    jl finish
    mov ecx,[C]
    mov [min],ecx.

finish:
    mov eax,answer
    call sprint

    mov eax, [min]
    call iprintLF

    call quit

```

Рис. 2.13: Программа task1.asm

```

d_kalashnikova@fedora:~/work/arch-pc/lab07$
d_kalashnikova@fedora:~/work/arch-pc/lab07$ nasm -f elf task1.asm
d_kalashnikova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 task1.o -o task1
d_kalashnikova@fedora:~/work/arch-pc/lab07$ ./task1
Input A: 24
Input B: 98
Input C: 15
Smallest: 15
d_kalashnikova@fedora:~/work/arch-pc/lab07$

```

Рис. 2.14: Запуск программы task1.asm

2. Напишите программу для вычисления функции $f(x)$ для введенных значений x и a :

Вариант 9:

$$f(x) = \begin{cases} x + a, & \text{если } x \leq a \\ a, & \text{если } x > a \end{cases}$$

При $x = 5, a = 7$ результат: 12.

При $x = 6, a = 4$ результат: 4.

Результаты программы представлены на рисунках (рис. 2.15, рис. 2.16).

```
task2.asm [----] 0 L: [ 18+28 46/ 49] *(633 /
mov edx,80
call sread
mov eax,A
call atoi.
mov [A],eax

mov eax,msgX
call sprint
mov ecx,X
mov edx,80
call sread.
mov eax,X
call atoi
mov [X],eax...

mov ebx, [X]
mov edx, [A]
cmp ebx, edx
jle first
jmp second

first:
mov eax,[X]
add eax,[A]
call iprintLF.
call quit
second:
mov eax,[A]
call iprintLF.
call quit
```

Рис. 2.15: Программа task2.asm

```
d_kalashnikova@fedora:~/work/arch-pc/lab07$  
d_kalashnikova@fedora:~/work/arch-pc/lab07$ nasm -f elf task2.asm  
d_kalashnikova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 task2.o -o task2  
d_kalashnikova@fedora:~/work/arch-pc/lab07$ ./task2  
Input A: 7  
Input X: 5  
12  
d_kalashnikova@fedora:~/work/arch-pc/lab07$ ./task2  
Input A: 4  
Input X: 6  
4  
d_kalashnikova@fedora:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы task2.asm

3 Выводы

Изучили команды условного и безусловного переходов, познакомились с фалом листинга.