

Отчёт по лабораторной работе

Лабораторная №7

Полина Витальевна Барабаш

Содержание

| | | |
|-----------------|---|------------------|
| <i>1</i> | <i>Цель работы</i> | <i>4</i> |
| <i>2</i> | <i>Выполнение работы</i> | <i>5</i> |
| <i>3</i> | <i>Выполнение самостоятельной работы</i> | <i>11</i> |
| <i>4</i> | <i>Выводы</i> | <i>13</i> |

Список иллюстраций

| | | |
|-----|--|----|
| 2.1 | Создание каталога и файла | 5 |
| 2.2 | Запуск программы с использованием jmp и результат её работы . | 6 |
| 2.3 | Изменение программы с использованием jmp и результат её работы | 6 |
| 2.4 | Самостоятельное изменение программы с инструкцией jmp и её работа | 7 |
| 2.5 | Создание файла, работа программы с условных переходом | 7 |
| 2.6 | Открытый после создания файл листинга | 8 |
| 2.7 | Создание файла листинга программы с ошибкой | 10 |
| 2.8 | Сообщение об ошибке в файле листинга | 10 |
| 3.1 | Работа программы по нахождению наименьшей из трех переменных | 11 |
| 3.2 | Вычисление данной функции с двумя вариантами переменных . | 12 |

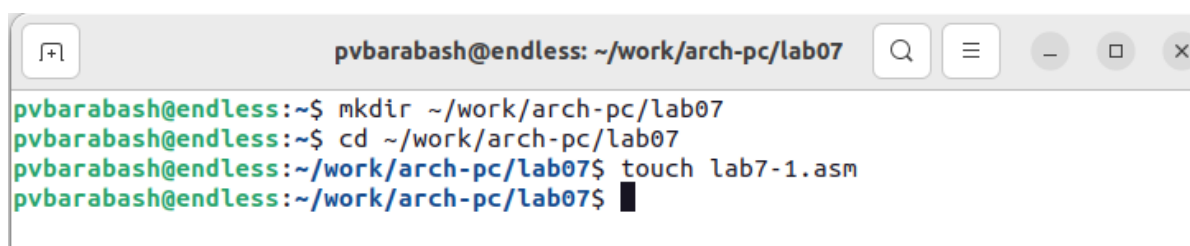
1 Цель работы

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

2 Выполнение работы

Задание №1. Создайте каталог для программ лабораторной работы № 7, перейдите в него и создайте файл lab7-1.asm

Я создала каталог для программ лабораторной работы № 7 с помощью команды `mkdir ~/work/arch-pc/lab07`. Затем я перешла в него с помощью команды `cd` и создала файл lab7-1.asm с помощью `touch` (рис. 2.1).

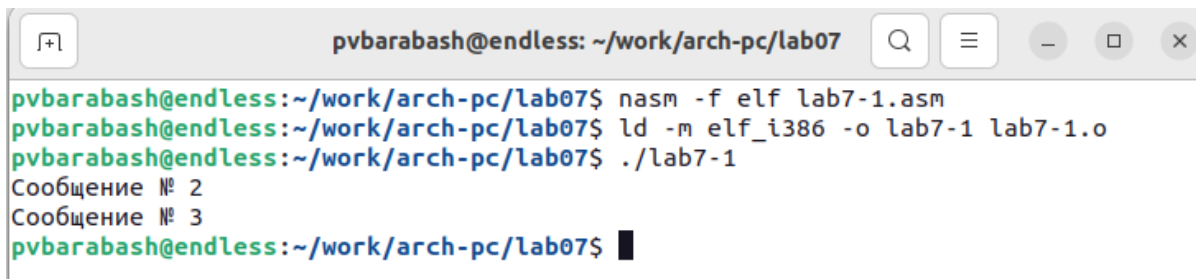


```
pvbarabash@endless: ~/work/arch-pc/lab07
pvbarabash@endless:~$ mkdir ~/work/arch-pc/lab07
pvbarabash@endless:~$ cd ~/work/arch-pc/lab07
pvbarabash@endless:~/work/arch-pc/lab07$ touch lab7-1.asm
pvbarabash@endless:~/work/arch-pc/lab07$
```

Рис. 2.1: Создание каталога и файла

Задание №2. Рассмотрите пример программы с использованием инструкции `jmp`. Введите в файл lab7-1.asm текст программы из листинга 7.1. Создайте исполняемый файл и запустите его.

Рассмотрев пример программы с использованием инструкции `jmp` (которая в NASM используется для реализации безусловных переходов), я ввела в файл lab7-1.asm текст программы из листинга 7.1. Затем я создала исполняемый файл и запустила его (рис. 2.2).



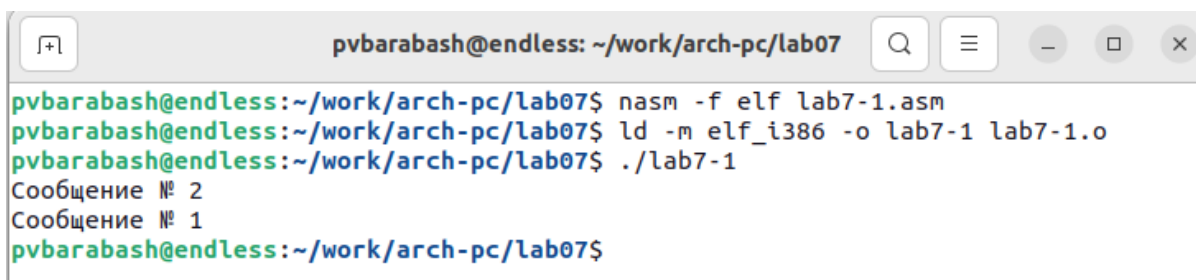
```
pvbarabash@endless: ~/work/arch-pc/lab07
pvbarabash@endless:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
pvbarabash@endless:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
pvbarabash@endless:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
pvbarabash@endless:~/work/arch-pc/lab07$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

Рис. 2.2: Запуск программы с использованием jmp и результат её работы

Результат работы программы такой же, как в руководстве по выполнению лабораторной работы.

Задание №3. Изменить текст программы в соответствии с листингом 7.2, чтобы после вывода “Сообщение № 2” выводилось “Сообщение № 1”, а затем программа завершалась. Создайте исполняемый файл и проверьте его работу.

Я изменила текст программы в соответствии с листингом 7.2. Я создала исполняемый файл и проверила его работу (рис. 2.3).



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

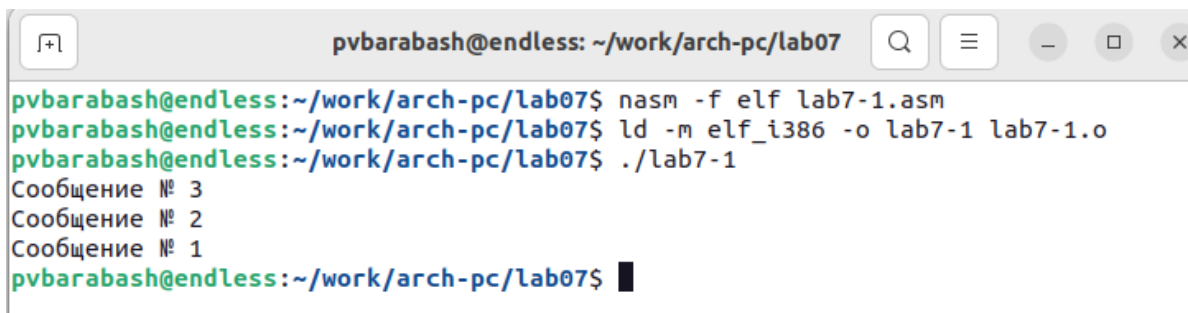
Рис. 2.3: Изменение программы с использованием jmp и результат её работы

Программа работает так, как ожидалось. Сначала выводит “Сообщение № 2”, а затем “Сообщение № 1” и завершает программу.

Задание №4. Измените текст программы добавив или изменив инструкции jmp, чтобы сначала выводилось “Сообщение № 3”, затем “Сообщение № 2”, затем “Сообщение № 1”.

Я изменила файл lab7-1.asm, изменив после _start: “jmp _label2” на “jmp _label3” и добавив инструкцию “jmp _label2” после выполнения _label3. Создала испол-

няемый файл и запустила его. Программа выдает необходимый результат (рис. 2.4).

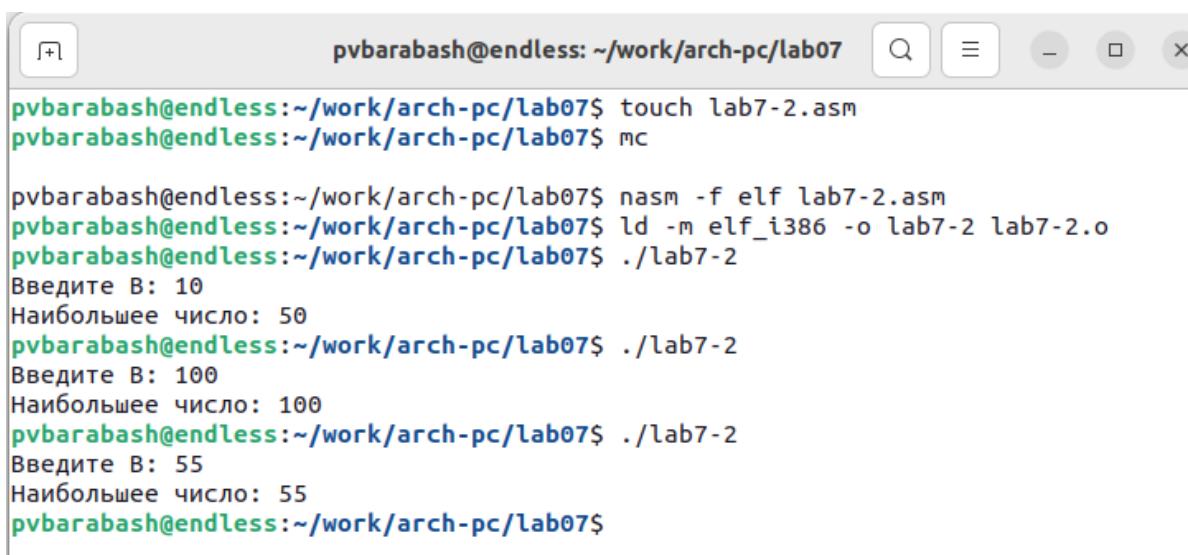


```
pvbarabash@endless: ~/work/arch-pc/lab07
pvbarabash@endless:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
pvbarabash@endless:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
pvbarabash@endless:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
pvbarabash@endless:~/work/arch-pc/lab07$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

Рис. 2.4: Самостоятельное изменение программы с инструкцией jmp и её работа

Задание №5. Создайте файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучите текст программы из листинга 7.3 и введите в lab7-2.asm. Создайте исполняемый файл и проверьте его работу для разных значений В.

Я создала файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 с помощью touch. Внимательно изучила текст программы из листинга 7.3 и ввела в lab7-2.asm. Я создала исполняемый файл и проверила его работу для разных значений В: 10, 100 и 55. Программа работает верно (рис. 2.5).



```
pvbarabash@endless: ~/work/arch-pc/lab07
pvbarabash@endless:~/work/arch-pc/lab07$ touch lab7-2.asm
pvbarabash@endless:~/work/arch-pc/lab07$ mc

pvbarabash@endless:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
pvbarabash@endless:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
pvbarabash@endless:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 10
Наибольшее число: 50
pvbarabash@endless:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 100
Наибольшее число: 100
pvbarabash@endless:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 55
Наибольшее число: 55
pvbarabash@endless:~/work/arch-pc/lab07$
```

Рис. 2.5: Создание файла, работа программы с условных переходом

Задание №6. Создайте файл листинга для программы из файла lab7-2.asm. Откройте файл листинга lab7-2.lst с помощью любого текстового редактора, например mcedit. Внимательно ознакомиться с его форматом и содержимым. Подробно объяснить содержимое трёх строк файла листинга по выбору.

Я создала файл листинга с помощью команды `nasm -f elf -l lab7-2.lst lab7-2.asm`. Затем я открыла файл листинга в помощью команды `mcedit lab7-2.lst` (рис. 2.6).

```

/home/pv~7-2.lst  [----] 59 L:[ 1+ 0 1/226] *(59 /13306b) 0109 0x06D [*][X]
1      %include 'in_out.asm'
2      <1> ;----- slen -----
3      <1> ; Функция вычисления длины сообщения
4      <1> slen:.....
5      00000000 53      <1>      push     ebx.....
6      00000001 89C3    <1>      mov      ebx, eax.....
7      <1>.....
8      <1> nextchar:.....
9      00000003 803800  <1>      cmp      byte [eax], 0...
10     00000006 7403    <1>      jz       finished.....
11     00000008 40      <1>      inc      eax.....
12     00000009 EBF8    <1>      jmp      nextchar.....
13     <1>.....
14     <1> finished:
15     0000000B 29D8    <1>      sub      eax, ebx
16     0000000D 5B      <1>      pop      ebx.....
17     0000000E C3      <1>      ret.....
18     <1>.....
19     <1>.....
20     <1> ;----- sprint -----
21     <1> ; Функция печати сообщения
22     <1> ; входные данные: mov eax,<message>
1Помощь 2Сох~ть 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход

```

Рис. 2.6: Открытый после создания файл листинга

Я внимательно ознакомилась с его форматом и содержимым. Я буду объяснять следующие строки:

18 000000F2 B9[0A000000] mov ecx,B

19 000000F7 BA0A000000 mov edx,10

20 000000FC E842FFFFFF

call sread

Этот кусок программы считывает введенное с клавиатуры число, записывает его в В. В правой части написаны сами строки кода. Левая часть состоит из номера строки, адреса и машинного кода. Адрес (000000F2, 000000F7, 000000FC) это смещение машинного кода от начала текущего сегмента. Начало текущего сегмента – начало программы, начиная с section .data. Машинный код это ассемблированная исходная строка в виде шестнадцатеричной последовательности.

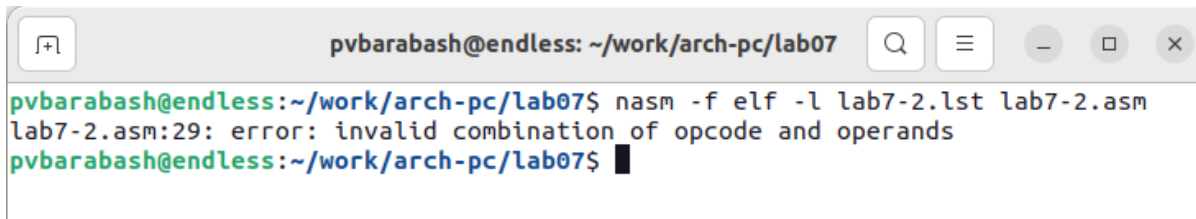
То есть инструкция mov esx,B начинается по смещению 000000F2 в сегменте кода; далее идет машинный код, в который ассемблируется инструкция, то есть инструкция mov esx,B ассемблируется в B9[0A000000], где B9 является mov esx, то есть запись в esx, а [0A000000] ссылается на B (то, что переносится в esx).

Инструкция mov edx,10 начинается по смещению 000000F7 в сегменте кода; далее идет машинный код BA0A000000, в который ассемблируется инструкция mov edx,10. Здесь BA является mov edx, то есть запись в edx, а 0A000000 указывает на 10, то есть на максимальную длину выводимой строки.

Инструкция call sread начинается по смещению 000000FC в сегменте кода; далее идет машинный код E842FFFFFF, в который ассемблируется инструкция call sread.

Задание №7. Откройте файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалить один операнд. Выполните трансляцию с получением файла листинга. Какие выходные файлы создаются в этом случае? Что добавляется в листинге?

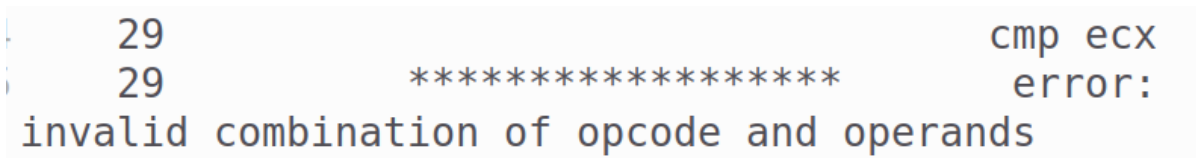
Я открыла файл с программой lab7-2.asm и в инструкции cmp esx,[C] удалила операнд [C]. Я выполнила трансляцию с получением файла листинга. В результате было выдано сообщение об ошибке (рис. 2.7).



```
pvbarabash@endless: ~/work/arch-pc/lab07
pvbarabash@endless:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:29: error: invalid combination of opcode and operands
pvbarabash@endless:~/work/arch-pc/lab07$
```

Рис. 2.7: Создание файла листинга программы с ошибкой

Файл листинга создаётся. На месте ошибочной инструкции также сообщается об ошибке (рис. 2.8).



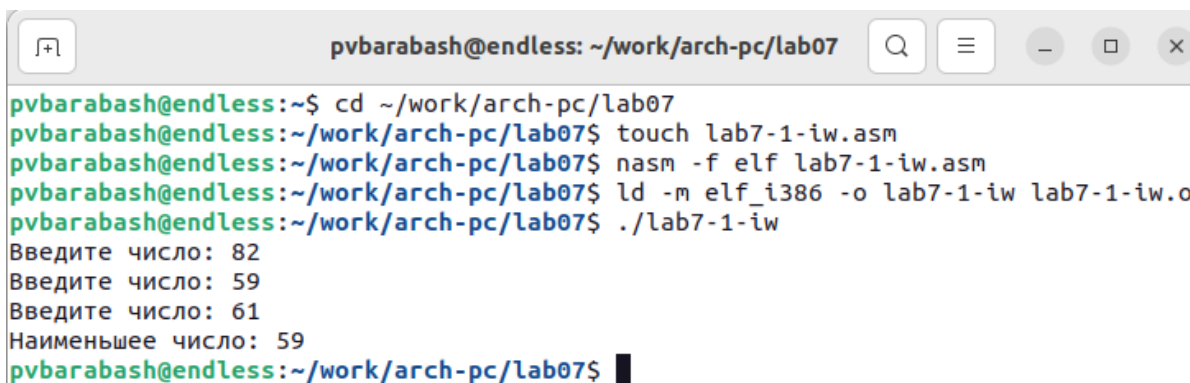
```
29                                     cmp ecx
29          ***** error:
invalid combination of opcode and operands
```

Рис. 2.8: Сообщение об ошибке в файле листинга

3 Выполнение самостоятельной работы

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 № 6. Создайте исполняемый файл и проверьте его работу.

В прошлой лабораторной работе был получен второй вариант. Следовательно, значения переменных 82, 59, 61. Я создала файл `lab7-1-iw.asm` и написала программу для нахождения наименьшей из 3 целочисленных переменных. Программа работает верно (рис. 3.1).



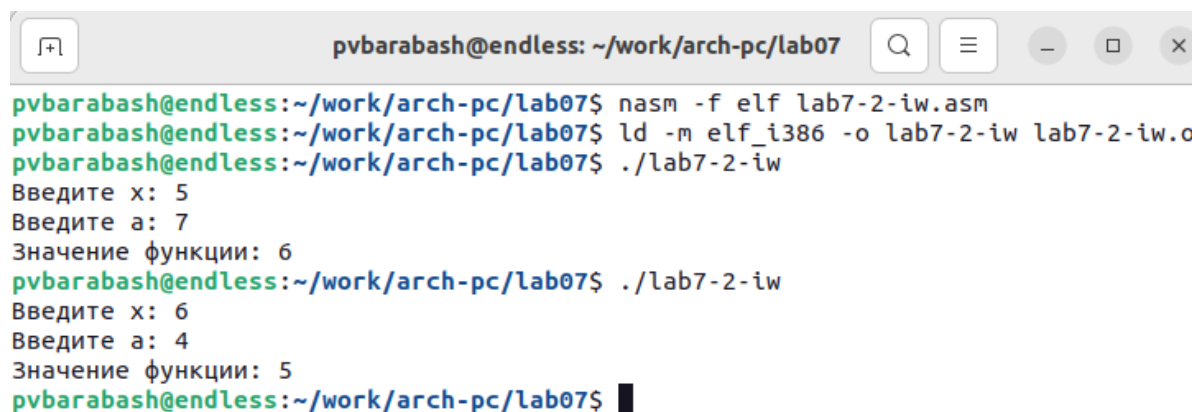
```
pvbarabash@endless: ~/work/arch-pc/lab07
pvbarabash@endless:~$ cd ~/work/arch-pc/lab07
pvbarabash@endless:~/work/arch-pc/lab07$ touch lab7-1-iw.asm
pvbarabash@endless:~/work/arch-pc/lab07$ nasm -f elf lab7-1-iw.asm
pvbarabash@endless:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1-iw lab7-1-iw.o
pvbarabash@endless:~/work/arch-pc/lab07$ ./lab7-1-iw
Введите число: 82
Введите число: 59
Введите число: 61
Наименьшее число: 59
pvbarabash@endless:~/work/arch-pc/lab07$
```

Рис. 3.1: Работа программы по нахождению наименьшей из трех переменных

Задание №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

№ 6. Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6.

Я создала файл `lab7-2-iw.asm` и написала программу для нахождения значения функции, данной во втором варианте. Если $x < a$, $f(x) = a - 1$. Если $a \geq x$, то $f(x) = x - 1$. Я проверила работу функции с данными значениями x и a . Программа работает верно (рис. 3.2).



```
pvbarabash@endless: ~/work/arch-pc/lab07
pvbarabash@endless:~/work/arch-pc/lab07$ nasm -f elf lab7-2-iw.asm
pvbarabash@endless:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2-iw lab7-2-iw.o
pvbarabash@endless:~/work/arch-pc/lab07$ ./lab7-2-iw
Введите x: 5
Введите a: 7
Значение функции: 6
pvbarabash@endless:~/work/arch-pc/lab07$ ./lab7-2-iw
Введите x: 6
Введите a: 4
Значение функции: 5
pvbarabash@endless:~/work/arch-pc/lab07$
```

Рис. 3.2: Вычисление данной функции с двумя вариантами переменных

4 Выводы

Я изучила команды условного и безусловного переходов, приобрела навыки написания программ с использованием переходов. Также я познакомилась с назначением и структурой файла листинга.