

Лабораторная работа номер 7

Отчёт

Виноградова Мария Андреевна

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 | 10 |
| 3.3 | Задание для самостоятельной работы | 13 |
| 4 | Выводы | 17 |

Список иллюстраций

| | | |
|------|--|----|
| 3.1 | Создаем каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code> | 7 |
| 3.2 | Заполняем файл | 7 |
| 3.3 | Запускаем файл и смотрим на его работу | 8 |
| 3.4 | Изменяем файл | 8 |
| 3.5 | Запускаем файл и смотрим на его работу | 8 |
| 3.6 | Редактируем файл | 9 |
| 3.7 | Проверяем, сошелся ли наш вывод с данным в условии выводом . | 9 |
| 3.8 | Создаем файл командой <code>touch</code> | 9 |
| 3.9 | Заполняем файл | 10 |
| 3.10 | Смотрим на работу программ | 10 |
| 3.11 | Создаем файл листинга | 11 |
| 3.12 | Изучаем файл | 11 |
| 3.13 | Удаляем операндум из файла | 12 |
| 3.14 | Транслируем файл | 12 |
| 3.15 | Изучаем файл с ошибкой | 13 |
| 3.16 | Создаем файл командой <code>touch</code> | 13 |
| 3.17 | Пишем программу | 14 |
| 3.18 | Смотрим на работу программы(всё верно) | 14 |
| 3.19 | Создаем файл командой <code>touch</code> | 15 |
| 3.20 | Пишем программу | 15 |
| 3.21 | Проверяем работу программы | 15 |
| 3.22 | Проверяем работу программы с другими переменными | 16 |

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

Написать программы для выбора наименьшего числа и решения системы выражений

3 Выполнение лабораторной работы

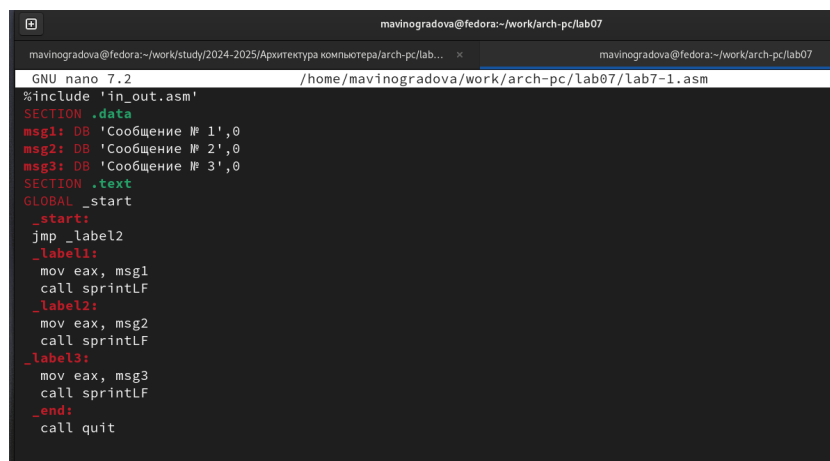
3.1 Реализация переходов в NASM

Создаем каталог для программ которые потребуются нам в ходе выполнения ЛБ7, и создаём в нём файл (рис. fig. 3.1).

```
mavinogradova@fedora:~$ mkdir ~/work/arch-pc/lab07
mavinogradova@fedora:~$ cd ^C
mavinogradova@fedora:~$ cd /work/arch-pc/lab07
bash: cd: /work/arch-pc/lab07: Нет такого файла или каталога
mavinogradova@fedora:~$ cd work
mavinogradova@fedora:~/work$ cd arch-pc
mavinogradova@fedora:~/work/arch-pc$ cd lab07
mavinogradova@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рис. 3.1: Создаем каталог с помощью команды `mkdir` и файл с помощью команды `touch`

Открываем файл в Midnight Commander и заполняем его так как показано в листинге 7.1 (рис. fig. 3.2).



```
mavinogradova@fedora:~/work/arch-pc/lab07
GNU nano 7.2 /home/mavinogradova/work/arch-pc/lab07/lab7-1.asm
%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
```

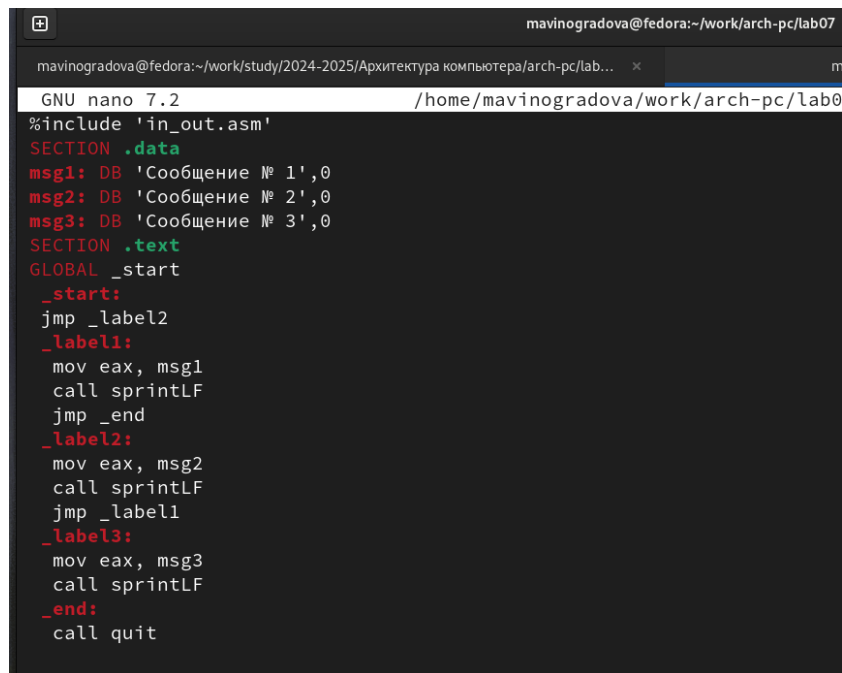
Рис. 3.2: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.3).

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

Рис. 3.3: Запускаем файл и смотрим на его работу

Снова открываем файл для редактирования и изменяем его так как показано в листинге 7.2 (рис. fig. 3.4).



```
mavinogradova@fedora:~/work/arch-pc/lab07
GNU nano 7.2 /home/mavinogradova/work/arch-pc/lab0
%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
```

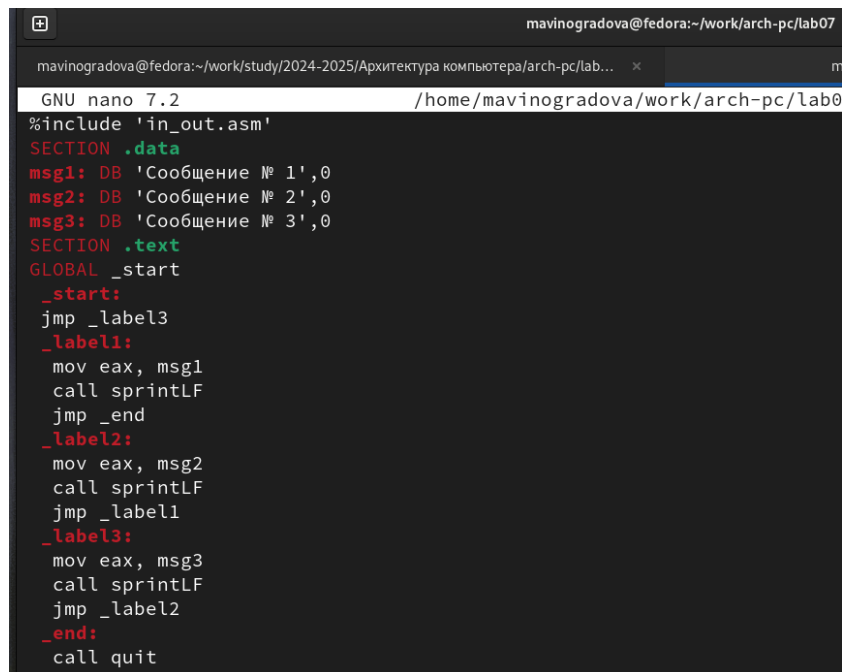
Рис. 3.4: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.5).

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

Рис. 3.5: Запускаем файл и смотрим на его работу

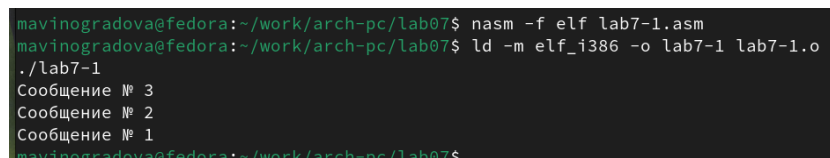
Снова открываем файл для редактирования и изменяем его так, чтобы полученный вывод совпал с заданным (рис. fig. 3.6).



```
GNU nano 7.2 /home/mavinogradova/work/arch-pc/lab0
%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. 3.6: Редактируем файл

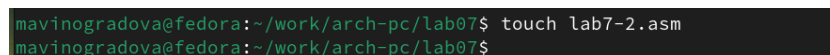
Создаем исполняемый файл и запускаем его (рис. fig. 3.7).



```
mavinogradova@fedora:~/work/arch-pc/lab0$ nasm -f elf lab7-1.asm
mavinogradova@fedora:~/work/arch-pc/lab0$ ld -m elf_i386 -o lab7-1 lab7-1.o
./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
mavinogradova@fedora:~/work/arch-pc/lab0$
```

Рис. 3.7: Проверяем, сошелся ли наш вывод с данным в условии выводом

Создаем новый файл (рис. fig. 3.8).



```
mavinogradova@fedora:~/work/arch-pc/lab0$ touch lab7-2.asm
mavinogradova@fedora:~/work/arch-pc/lab0$
```

Рис. 3.8: Создаем файл командой touch

Открываем файл в Midnight Commander и заполняем его так как показано в листинге 7.3 (рис. fig. 3.9).

```
mavinogradova@fedora:~/work/arch-pc/lab07
GNU nano 7.2 /home/mavinogradova/work/arch-pc/lab07/lab7-
#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] ; 'ecx = A'
    mov [max],ecx ; 'max = A'

    cmp ecx,[C] ; Сравниваем 'A' и 'C'
    jg check_B ; если 'A>C', то переход на метку 'check_B',
    mov ecx,[C] ; иначе 'ecx = C'
    mov [max],ecx ; 'max = C'

check_B:
```

Рис. 3.9: Заполняем файл

Создаем исполняемый файл и проверяем его работу, вводя разные значения B (рис. fig. 3.10).

```
mavinogradova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
mavinogradova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
mavinogradova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
mavinogradova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
mavinogradova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
mavinogradova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
mavinogradova@fedora:~/work/arch-pc/lab07$
```

Рис. 3.10: Смотрим на работу программ

3.2 Изучение структуры файлы листинга

Создаем файл листинга для программы lab7-2.asm (рис. fig. 3.11).

```
mavinogradova@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
mavinogradova@fedora:~/work/arch-pc/lab07$ mcedit lab7-2.lst
```

Рис. 3.11: Создаем файл листинга

Открываем файл листинга с помощью команды `mcedit` и изучаем его (рис. fig. 3.12).

```
lab7-2.lst  [----]  0 L: 1+ 0 1/224] *(0 /13698b) 0032 0x020
1  %include 'in_out.asm'
2  <1> ;----- slen -----
3  <1> ; Функция вычисления длины сообщения
4  <1> slen:.....
5  <1>   push    ebx.....
6  <1>   mov     ebx, eax.....
7  <1>   .....
8  <1> nextchar:.....
9  <1>   cmp     byte [eax], 0...
10 <1>   jz      finished.....
11 <1>   inc     eax.....
12 <1>   jmp     nextchar.....
13 <1> finished:
14 <1>   sub     eax, ebx
15 <1>   pop     ebx.....
16 <1>   ret.....
17 <1>
18 <1>
19 <1> ;----- sprint -----
20 <1> ; Функция печати сообщения
21 <1> ; входные данные: mov eax,<message>
22 <1> sprint:
23 <1>   push    edx
24 <1>   push    ecx
25 <1>   push    ebx
26 <1>   push    eax
27 <1>   call    slen
28 <1>   .....
29 <1>   mov     edx, eax
30 <1>   pop     eax
31 <1>   .....
32 <1>   mov     ecx, eax
33 <1>   mov     ebx, 1
```

Рис. 3.12: Изучаем файл

Строка 45: `BV00000000 - 0000001D` - адрес в сегменте кода, `BV00000000` - машинный код, `mov ebx, 0` - присвоение переменной `ebx` значения 0

Строка 52: `59 - 0000002D` - адрес в сегменте кода, `59` - машинный код, `pop ecx` - восстановление значения регистра `ecx` из стека.

Строка 88: `F7FE - 00000042` - адрес в сегменте кода, `F7FE` - машинный код, `idiv esi` - целочисленное деление значения в регистре `eax` на значение в регистре `esi`.

Открываем файл и удаляем один операндум (рис. fig. 3.13).

```

section             .text
    global _start
_start:
    mov eax,msg1
    call sprint

    mov ecx,B
    mov edx,
    call sread

```

Рис. 3.13: Удаляем операндум из файла

Создаем исполняемый файл и запускаем его с получением файла листинга (рис. fig. 3.14).

```

mavinogradova@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.a
lab7-2.asm:17: error: invalid combination of opcode and operands
mavinogradova@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2  lab7-2.asm  lab7-2.lst
mavinogradova@fedora:~/work/arch-pc/lab07$

```

Рис. 3.14: Транслируем файл

При исполнении файла, выдается ошибка, но создаются исполнительный файл lab7-2 и lab7-2.lst

Снова открываем файл листинга и изучаем его (рис. fig. 3.15).

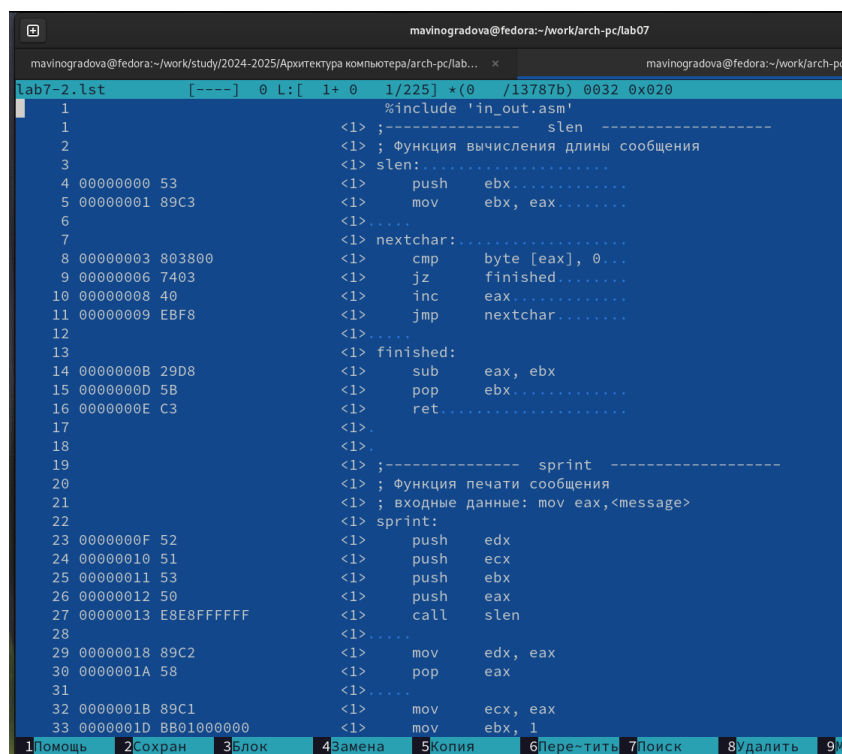


Рис. 3.15: Изучаем файл с ошибкой

3.3 Задание для самостоятельной работы

ВАРИАНТ-12

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.

Создаем новый файл (рис. fig. 3.16).

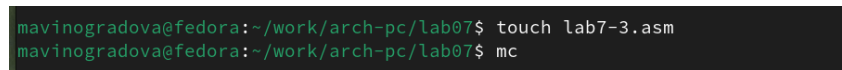


Рис. 3.16: Создаем файл командой touch

Открываем созданный файл и пишем программу, которая выберет наименьшее число из трех (2 числа уже в программе, 3-е вводится из консоли) (рис. fig. 3.17).

```
mavinogradova@fedora:~/work/arch-pc/lab07
GNU nano 7.2 /home/mavinogradova/work/arch-pc/lab07/lab7-3.asm
%include 'in_out.asm'
section .data
    msg1 db 'Введите B: ',0h
    msg2 db "Наименьшее число: ",0h
    A dd '99'
    C dd '26'
section .bss
    min 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 [min],ecx

    cmp ecx,[C]
    jl check_B
    mov ecx,[C]
    mov [min],ecx

check_B:
```

Рис. 3.17: Пишем программу

Создаем исполняемый файл, запускаем его и смотрим на работу программы (рис. fig. 3.18).

```
mavinogradova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
mavinogradova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
mavinogradova@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 29
Наименьшее число: 26
mavinogradova@fedora:~/work/arch-pc/lab07$
```

Рис. 3.18: Смотрим на работу программы(всё верно)

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.

Создаем новый файл (рис. fig. 3.19).

```
mavinogradova@fedora:~/work/arch-pc/lab07$ touch lab7-4.asm
mavinogradova@fedora:~/work/arch-pc/lab07$
```

Рис. 3.19: Создаем файл командой touch

Открываем файл и пишем программу, которая решит систему выражений из варианта номер которого получен в ходе выполнения лабораторной (12) (рис. fig. 3.20).

```
%include 'in_out.asm'
SECTION .data
    msg1: DB 'Введите x: ', 0h
    msg2: DB 'Введите a: ', 0h
    otv: DB 'F(x) = ', 0h

SECTION .bss
    x: RESB 80
    a: RESB 80
    res: RESB 80

SECTION .text
    GLOBAL _start

_start:
    mov eax, msg1
    call sprint

    mov ecx, x
    mov edx, 80
    call sread

    mov eax, x
    call atoi
    mov [x], eax

    mov eax, msg2
    call sprint

    mov ecx, a
    mov edx, 80
    call sread

    mov eax, a
```

Рис. 3.20: Пишем программу

Создаем исполняемый файл, запускаем его и проверяем его работу при $x=3$ и $a=7$ (рис. fig. 3.21).

```
mavinogradova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
mavinogradova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
mavinogradova@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 3
Введите a: 7
F(x) = 21
mavinogradova@fedora:~/work/arch-pc/lab07$
```

Рис. 3.21: Проверяем работу программы

Создаем исполняемый файл, запускаем его и проверяем его работу при $x=6$ и $a=4$ (рис. fig. 3.22).

```
mavinogradova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
mavinogradova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
mavinogradova@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 6
Введите a: 4
F(x) = 1
mavinogradova@fedora:~/work/arch-pc/lab07$
```

Рис. 3.22: Проверяем работу программы с другими переменными

4 Выводы

При выполнении лабораторной работы я изучила команды условных и безусловных переходов, а также приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файлов листинга.