

Лабораторная работа №7

Архитектура компьютера

Сафиуллина Айлина Саяровна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация переходов в NASM	8
4.2	Изучение структуры файлов листинга	9
4.3	Задание для самостоятельной работы	14
5	Выводы	17

Список иллюстраций

4.1	создание файла lab7-1.asm	8
4.2	листинг 7.1	9
4.3	проверка работы исполняемого файла	9
4.4	результат программы	9
4.5	листинг 7.2	10
4.6	результат программы	10
4.7	замена инструкции jmp	11
4.8	замена инструкции jmp	11
4.9	создание файла lab7-2.asm	12
4.10	листинг 7.3	12
4.11	проверка работы исполняемого файла	12
4.12	использование ключа -l	13
4.13	листинг в mscedit	13
4.14	проверка программы	14
4.15	вариант по студенческому	14
4.16	задание для самостоятельной работы п.1	15
4.17	листинг для самостоятельной работы п.2	16
4.18	проверка работы листинга для с.р. п.2	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

1. Реализация переходов в NASM
2. Изучение структуры файлов листинга
3. Задание для самостоятельной работы

3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

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

(рис. 4.1)

```
assafiullina@dk8n78 ~ $ mkdir ~/work/arch-pc/lab07
assafiullina@dk8n78 ~ $ cd ~/work/arch-pc/lab07
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ touch lab7-1.asm
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ █
```

Рис. 4.1: создание файла lab7-1.asm

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.

(рис. 4.2)


```

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение № 1'
13 _label2:
14 mov eax, msg2 ; Вывод на экран строки
15 call sprintf ; 'Сообщение № 2'
16 Архитектура ЭВМ
17 _label3:
18 mov eax, msg3 ; Вывод на экран строки
19 call sprintf ; 'Сообщение № 3'
20 _end:
21 call quit ; вызов подпрограммы завершения

```

Рис. 4.2: листинг 7.1

4.2 Изучение структуры файлов листинга

Создадим исполняемый файл и проверим его работу.

(рис. 4.3)

```

assafiullina@dk8n78 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ ./lab7-1

```

Рис. 4.3: проверка работы исполняемого файла

Сравним результат работы данной программы с приведенным в методическом пособии

(рис. 4.4)

```

assafiullina@dk8n78 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
assafiullina@dk8n78 ~/work/arch-pc/lab07 $

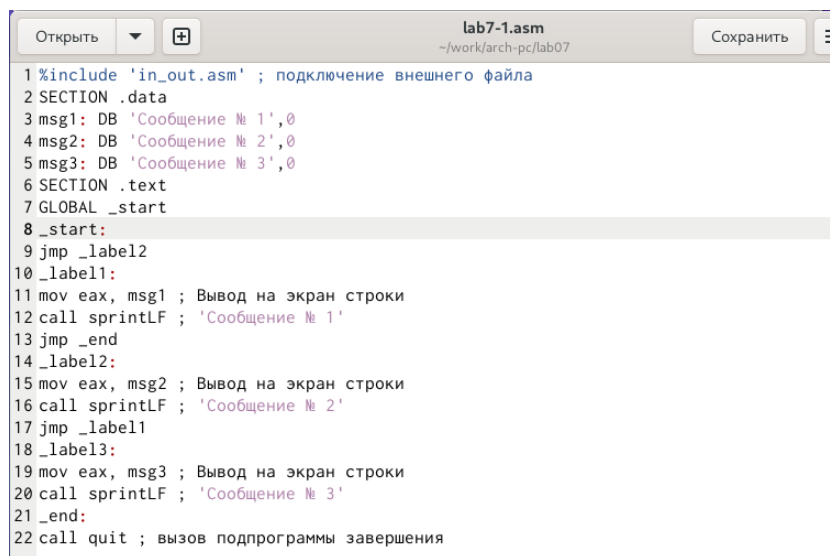
```

Рис. 4.4: результат программы

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.

(рис. 4.5)



```

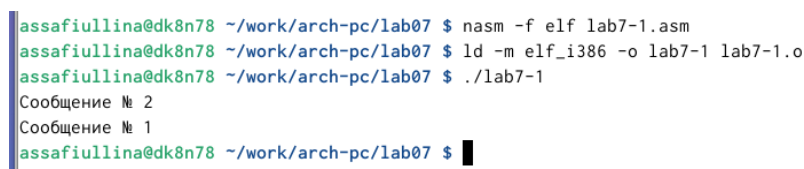
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call printf ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call printf ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call printf ; 'Сообщение № 3'
21 _end:
22 call quit ; вызов подпрограммы завершения

```

Рис. 4.5: листинг 7.2

Создадим исполняемый файл и проверим его работу

(рис. 4.6)



```

assafiullina@dk8n78 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
assafiullina@dk8n78 ~/work/arch-pc/lab07 $

```

Рис. 4.6: результат программы

Изменим текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим:

assafiullina@dk8n78:~\$./lab7-1 Сообщение № 3 Сообщение № 2 Сообщение № 1
1 assafiullina@dk8n78:~\$

Для этого в 9 строке заменим `jmp _label2` на `jmp _label3`

(рис. 4.7)

```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label3
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintf ; 'Сообщение № 3'
21 jmp _label2
22 _end:
23 call quit ; вызов подпрограммы завершения
```

Рис. 4.7: замена инструкции `jmp`

Проверим, соответствует ли результат нашей программы заданному условию, создав исполняемый файл

(рис. 4.8)

```
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
assafiullina@dk8n78 ~/work/arch-pc/lab07 $
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintf ; 'Сообщение № 3'
21 jmp _label2
22 _end:
23 call quit ; вызов подпрограммы завершения
```

Рис. 4.8: замена инструкции `jmp`

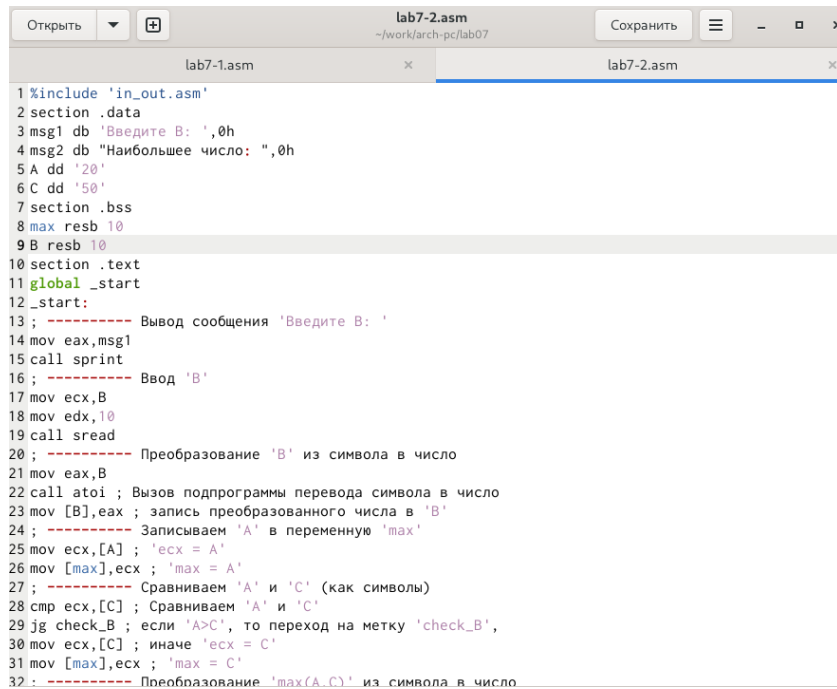
Создадим файл `lab7-2.asm` в каталоге `~/work/arch-pc/lab07`.

(рис. 4.9)

```
assafiullina@dk8n78 ~ $ cd ~/work/arch-pc/lab07
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ touch lab7-2.asm
assafiullina@dk8n78 ~/work/arch-pc/lab07 $
```

Рис. 4.9: создание файла lab7-2.asm

Изучим текст программы из листинга 7.3 и введем в lab7-2.asm.
(рис. 4.10)



```
lab7-2.asm
1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '20'
6 C dd '50'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод 'B'
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,B
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A] ; 'ecx = A'
26 mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 jg check_B ; если 'A>C', то переход на метку 'check_B',
30 mov ecx,[C] ; иначе 'ecx = C'
31 mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
```

Рис. 4.10: листинг 7.3

Создадим исполняемый файл и проверим его работу для разных значений
(рис. 4.11)

```
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 2
Наибольшее число: 50
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 1000
Наибольшее число: 1000
assafiullina@dk8n78 ~/work/arch-pc/lab07 $
```

Рис. 4.11: проверка работы исполняемого файла

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` (рис. 4.12)

```
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рис. 4.12: использование ключа `-l`

Откроем файл листинга `lab7-2.lst` с помощью текстового редактора, `mcedit`:
`mcedit lab7-2.lst`

(рис. 4.13)

```
lab7-2.lst  [----]  0 L: 1+ 0 1/225] *(0 /14458b) 0032 0x020 [*][X]
1          %include 'in_out.asm'
2          <1> ;----- slen -----
3          <1> ; Функция вычисления длины сообщения
4          <1> slen:.....
5          4 00000000 53          <1> push    ebx.....
6          5 00000001 89C3       <1> mov     ebx, eax.....
7          6          <1>.....
8          7          <1> nextchar:.....
9          8 00000003 803800     <1> cmp     byte [eax], 0...
10         9 00000006 7403       <1> jz      finished.....
11        10 00000008 40         <1> inc     eax.....
12        11 00000009 EBF8       <1> jmp     nextchar.....
13        12          <1>.....
14        13          <1> finished:
15        14 0000000B 29D8       <1> sub     eax, ebx
16        15 0000000D 5B         <1> pop     ebx.....
17        16 0000000E C3         <1> ret.....
18        17          <1>.....
19        18          <1>.....
20        19          <1> ;----- sprint -----
21        20          <1> ; Функция печати сообщения
22        21          <1> ; входные данные: mov eax,<message>
```

Рис. 4.13: листинг в `mcedit`

Откроем файл с программой `lab7-2.asm` и удалим один операнд - `max` в строке 38. Выполним трансляцию с получением файла листинга: `nasm -f elf -l lab7-2.lst lab7-2.asm`

(рис. 4.14)

Рис. 4.14: проверка программы

4.3 Задание для самостоятельной работы

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.

(рис. 4.15)

```
assafiullina@dk8n78 ~/work/arch-pc/lab06 $ touch variant.asm
assafiullina@dk8n78 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
assafiullina@dk8n78 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
assafiullina@dk8n78 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1032241171
Ваш вариант: 12
```

Рис. 4.15: вариант по студенческому

(рис. ??)

```

lab7-1.asm x lab7-2.asm x lab6-3.asm x variant.asm x lab7-3.asm x
1 %include 'in_out.asm'
2 section .data
3 msg1 db 'a=99,c=26 , введите b: ',0h
4 msg2 db "Наименьшее число: ",0h
5 A dd '99'
6 C dd '26'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 mov eax,msg1
14 call sprint
15 mov ecx,B
16 mov edx,10
17 call sread
18 mov eax,B
19 call atoi
20 mov [B],eax
21 mov ecx,[A] ; 'ecx = A'
22 mov [max],ecx ; 'max = A'
23 cmp ecx,[C] ; Сравниваем 'A' и 'C'
24 jl check_B ; если 'A<C', то переход на метку 'check_B',
25 mov ecx,[C] ; иначе 'ecx = C'
26 mov [max],ecx ; 'max = C'
27 check_B:
28 mov eax,max
29 call atoi ; Вызов подпрограммы перевода символа в число
30 mov [max],eax ; запись преобразованного числа в max
31 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)

```

Создадим исполняемый файл и проверим его работу.

(рис. 4.16)

```

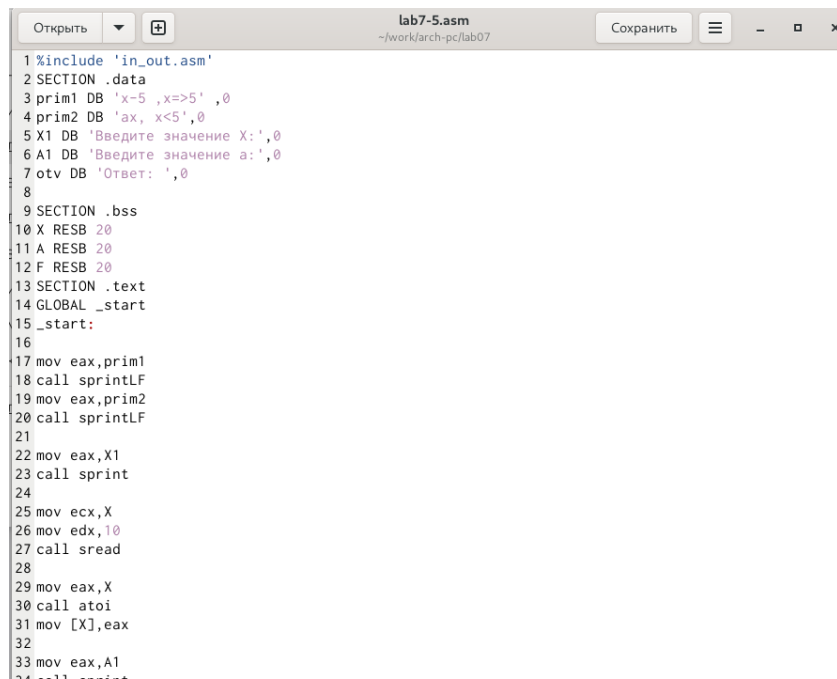
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ ./lab7-3
a=99,c=26 , введите b: 45
Наименьшее число: 26
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ ./lab7-3
a=99,c=26 , введите b: 29
Наименьшее число: 26
assafiullina@dk8n78 ~/work/arch-pc/lab07 $ █

```

Рис. 4.16: задание для самостоятельной работы п.1

Напишем программу, которая для введенных с клавиатуры значений \square и \square вычисляет значение заданной функции $\square(\square)$ и выводит результат вычислений. Вид функции $\square(\square)$ выберем из таблицы 7.6 вариантов заданий в соответствии с вариантом (12), полученным при выполнении лабораторной работы №6.

(рис. 4.17)



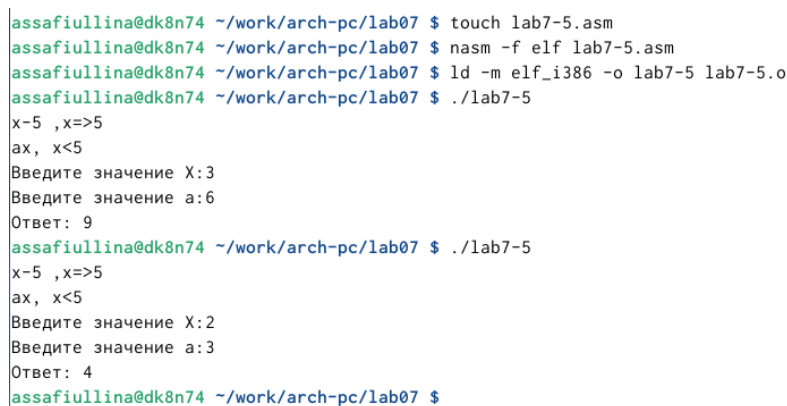
```

1 %include 'in_out.asm'
2 SECTION .data
3 prim1 DB 'x-5 ,x=>5' ,0
4 prim2 DB 'ax, x<5',0
5 X1 DB 'Введите значение X:',0
6 A1 DB 'Введите значение a:',0
7 otv DB 'Ответ: ',0
8
9 SECTION .bss
10 X RESB 20
11 A RESB 20
12 F RESB 20
13 SECTION .text
14 GLOBAL _start
15 _start:
16
17 mov eax,prim1
18 call sprintLF
19 mov eax,prim2
20 call sprintLF
21
22 mov eax,X1
23 call sprint
24
25 mov ecx,X
26 mov edx,10
27 call sread
28
29 mov eax,X
30 call atoi
31 mov [X],eax
32
33 mov eax,A1

```

Рис. 4.17: листинг для самостоятельной работы п.2

Создадим исполняемый файл и проверьте его работу для значений \square и \square из 7.6.
(рис. 4.18)



```

assafiullina@dk8n74 ~/work/arch-pc/lab07 $ touch lab7-5.asm
assafiullina@dk8n74 ~/work/arch-pc/lab07 $ nasm -f elf lab7-5.asm
assafiullina@dk8n74 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-5 lab7-5.o
assafiullina@dk8n74 ~/work/arch-pc/lab07 $ ./lab7-5
x-5 ,x=>5
ax, x<5
Введите значение X:3
Введите значение a:6
Ответ: 9
assafiullina@dk8n74 ~/work/arch-pc/lab07 $ ./lab7-5
x-5 ,x=>5
ax, x<5
Введите значение X:2
Введите значение a:3
Ответ: 4
assafiullina@dk8n74 ~/work/arch-pc/lab07 $

```

Рис. 4.18: проверка работы листинга для с.р. п.2

5 Выводы

В ходе данной лабораторной работы мы изучили команды условного и безусловного переходов. Приобрели навыки написания программ с использованием переходов. Познакомились с назначением и структурой файла листинга. А также выполнили задания для самостоятельной работы.