

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук  
Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7

дисциплина:     Архитектура компьютера

Студент: Крыловецкий Денис Витальевич

Группа: НКАбд-03-25

МОСКВА

2025 г.

## Оглавление

<u>Список таблиц</u>	1	Цель работы	4
<u>2 Задания</u>	6		
<u>3 Теоретическое введение</u>	7		
<u>4 Выполнение работы</u>	8		
<u>Реализация переходов в NASM</u>	8		
<u>Изучение структуры файлы листинга</u>	13		
<u>Объяснение строк листинга</u>	13		
<u>Задание для самостоятельной работы</u>	15		
<u>5 Выводы</u>	19		
<u>Список литературы</u>	20		

## Список изображений

<a href="#">Рисунок 0.1 (создание каталога)</a> .....	7
<a href="#">Рисунок 0.2 (переход в каталог)</a> .....	7
<a href="#">Рисунок 0.3 (создаем файл)</a> .....	7
<a href="#">Рисунок 0.4 (текст программы)</a> .....	7
<a href="#">Рисунок 0.5 (создание исполняемого файла)</a> .....	8
<a href="#">Рисунок 0.6 (запуск исполняемого файла)</a> .....	8
<a href="#">Рисунок 0.7 (результат работы файла)</a> .....	8
<a href="#">Рисунок 0.8 (измененный текст программы)</a> .....	8
<a href="#">Рисунок 0.9 (создание и запуск исполняемого файла)</a> .....	9
<a href="#">Рисунок 0.10 (измененный текст программы)</a> .....	9
<a href="#">Рисунок 0.11 (создание исполняемого файла)</a> .....	9
<a href="#">Рисунок 0.12(запуск исполняемого файла)</a> .....	10
<a href="#">Рисунок 0.13(создание файла)</a> .....	10
<a href="#">Рисунок 0.14 (текст программы)</a> .....	10
<a href="#">Рисунок 0.15 (текст программы)</a> .....	11
<a href="#">Рисунок 0.16 (создание исполняемого файла и проверка работы)</a> .....	11
<a href="#">Рисунок 0.17(создание исполняемого файла и проверка работы)</a> .....	11
<a href="#">Рисунок 0.18(создание исполняемого файла и проверка работы)</a> .....	11
<a href="#">Рисунок 0.19 (создание файла листинга)</a> .....	12
<a href="#">Рисунок 0.20 (открытие листинга)</a> .....	12
<a href="#">Рисунок 0.21 (открытие листинга)</a> .....	12
<a href="#">Рисунок 0.22 (текст программы с удалением операнды)</a> .....	13
<a href="#">Рисунок 0.23 (создание файла листинга)</a> .....	13
<a href="#">Рисунок 0.24 (открытие листинга)</a> .....	13
<a href="#">Рисунок 0.25 (текст программы)</a> .....	14
<a href="#">Рисунок 0.26 (текст программы)</a> .....	14
<a href="#">Рисунок 0.27 (текст программы)</a> .....	14
<a href="#">Рисунок 0.28 (создание исполняемого файла и проверка его работы)</a> .....	15
<a href="#">Рисунок 0.29 (текст программы)</a> .....	15
<a href="#">Рисунок 0.30 (текст программы)</a> .....	16
<a href="#">Рисунок 0.31 (создание исполняемого файла)</a> .....	16
<a href="#">Рисунок 0.32 (создание исполняемого файла)</a> .....	16
<a href="#">Рисунок 0.33 (запуск программы)</a> .....	16
<a href="#">Рисунок 0.34 (проверка результатов)</a> .....	16
<a href="#">Рисунок 0.35 (проверка результатов)</a> .....	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 Задания**

1. Реализация переходов в NASM
2. Изучение структуры файлов листинга
3. Самостоятельное написание программ по материалам лабораторной работы.

### 3 Теоретическое введение

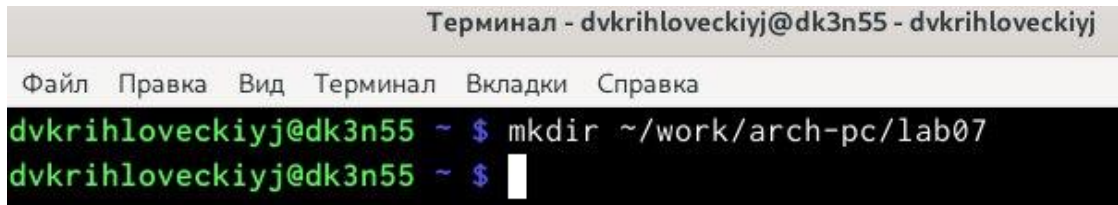
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- 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 работы

### Реализация переходов в NASM

Создадим каталог для программ лабораторной работы № 7, перейдем в него и создадим файл lab7-1.asm (рисунок 0.1-0.3)



```
Терминал - dvkrihloveckiyj@dk3n55 - dvkrihloveckiyj
Файл  Правка  Вид  Терминал  Вкладки  Справка
dvkrihloveckiyj@dk3n55 ~ $ mkdir ~/work/arch-pc/lab07
dvkrihloveckiyj@dk3n55 ~ $
```

Рисунок 0.1 (создание каталога)



```
dvkrihloveckiyj@dk3n55 ~ $ mkdir ~/work/arch-pc/lab07
dvkrihloveckiyj@dk3n55 ~ $ cd ~/work/arch-pc/lab07
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $
```

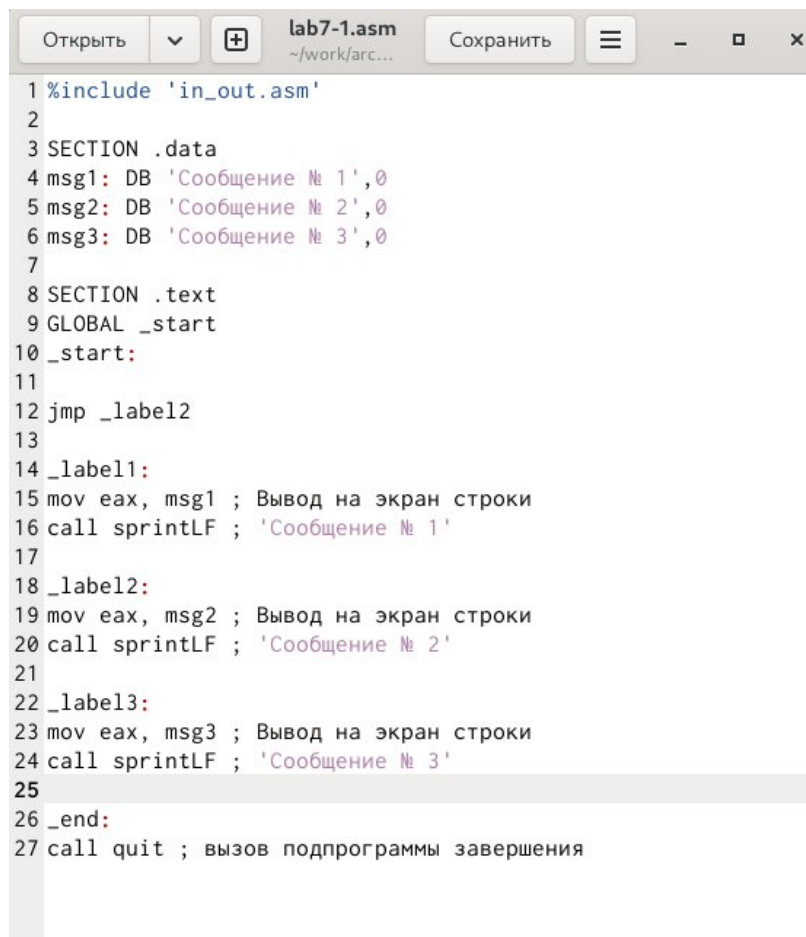
Рисунок 0.2 (переход в каталог)



```
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ touch lab7-1.asm
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.3 (создаем файл)

Введем в файл lab7-1.asm текст программы из листинга (рисунок 0.4)



```
Открыть  ▾  [+  lab7-1.asm  ~/work/arc...  Сохранить  ≡  -  □  x
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1',0
5 msg2: DB 'Сообщение № 2',0
6 msg3: DB 'Сообщение № 3',0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label2
13
14 _label1:
15 mov eax, msg1 ; Вывод на экран строки
16 call sprintLF ; 'Сообщение № 1'
17
18 _label2:
19 mov eax, msg2 ; Вывод на экран строки
20 call sprintLF ; 'Сообщение № 2'
21
22 _label3:
23 mov eax, msg3 ; Вывод на экран строки
24 call sprintLF ; 'Сообщение № 3'
25
26 _end:
27 call quit ; вызов подпрограммы завершения
```

Рисунок 0.4 (текст программы)

Создадим исполняемый файл и запустим его (рисунок 0.5-0.6)

```
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.5 (создание исполняемого файла)

```
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
```

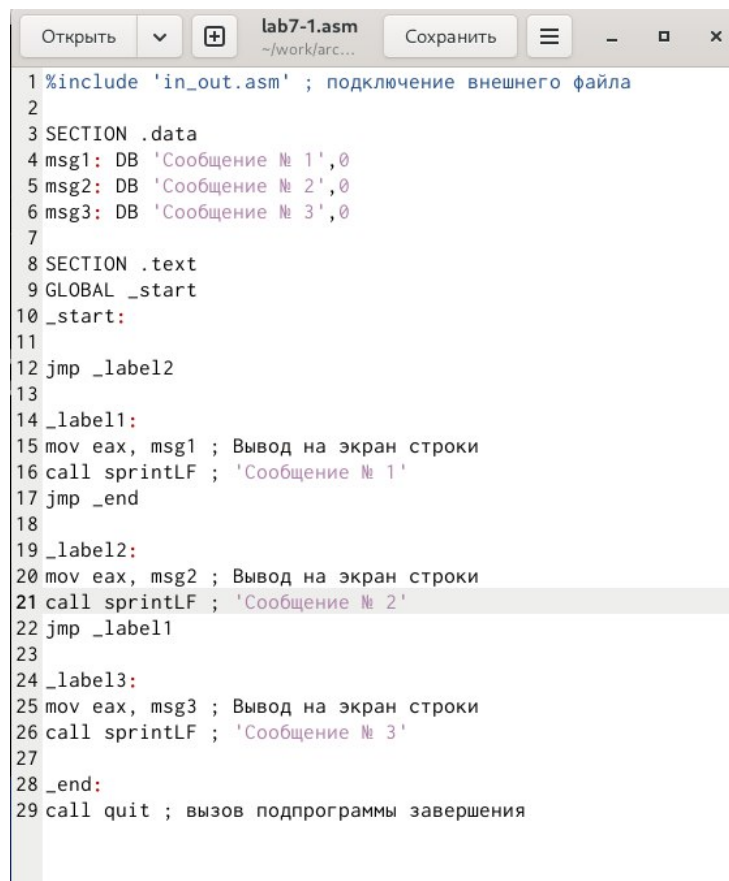
Рисунок 0.6 (запуск исполняемого файла)

Результат работы данной программы будет следующим (рисунок 0.7)

```
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.7 (результат работы файла)

Изменим текст программы в соответствии с листингом (рисунок 0.8)



```
Открыть  lab7-1.asm  Сохранить
~/work/arch...
1 %include 'in_out.asm' ; подключение внешнего файла
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1',0
5 msg2: DB 'Сообщение № 2',0
6 msg3: DB 'Сообщение № 3',0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label2
13
14 _label1:
15 mov eax, msg1 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 1'
17 jmp _end
18
19 _label2:
20 mov eax, msg2 ; Вывод на экран строки
21 call sprintf ; 'Сообщение № 2'
22 jmp _label1
23
24 _label3:
25 mov eax, msg3 ; Вывод на экран строки
26 call sprintf ; 'Сообщение № 3'
27
28 _end:
29 call quit ; вызов подпрограммы завершения
```

Рисунок 0.8 (измененный текст программы)

Создадим исполняемый файл и проверим его работу (рисунок 0.9)

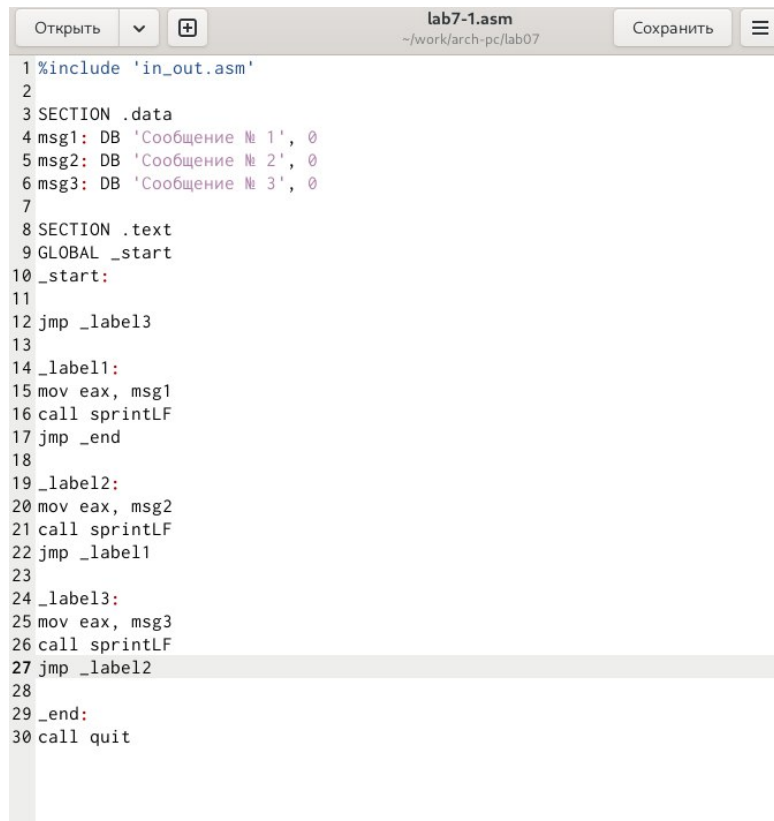
```

dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ gedit lab7-1.asm
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $

```

Рисунок 0.9 (создание и запуск исполняемого файла)

Изменим текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим (рисунок 0.10)



```

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1', 0
5 msg2: DB 'Сообщение № 2', 0
6 msg3: DB 'Сообщение № 3', 0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label3
13
14 _label1:
15 mov eax, msg1
16 call sprintf
17 jmp _end
18
19 _label2:
20 mov eax, msg2
21 call sprintf
22 jmp _label1
23
24 _label3:
25 mov eax, msg3
26 call sprintf
27 jmp _label2
28
29 _end:
30 call quit

```

Рисунок 0.10 (измененный текст программы)

```

dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ gedit lab7-1.asm
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o

```

Рисунок 0.11 (создание исполняемого файла)

```

dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $

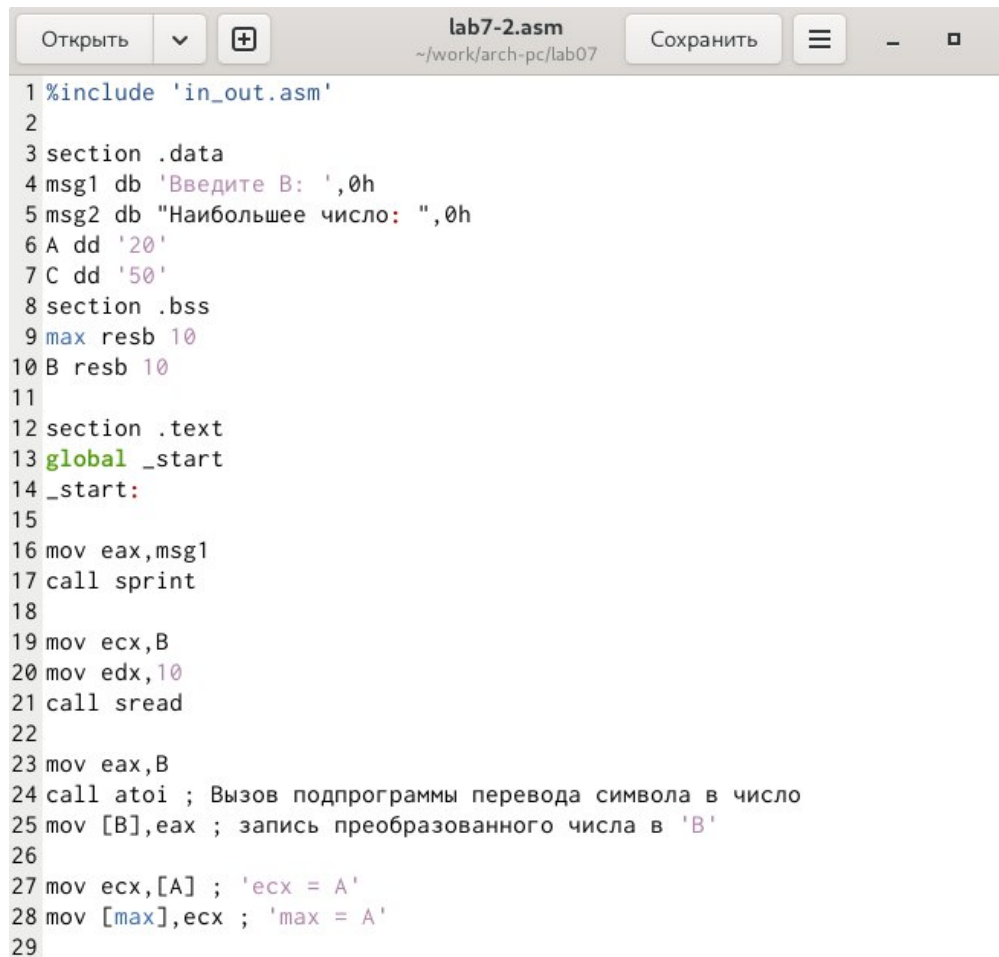
```

Рисунок 0.12(запуск исполняемого файла)

Создадим файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучим текст программы из листинга 7.3 и введем в lab7-2.asm (рисунок 0.13-0.15)

```
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ touch lab7-2.asm
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.13(создание файла)



```
1 %include 'in_out.asm'
2
3 section .data
4 msg1 db 'Введите B: ',0h
5 msg2 db "Наибольшее число: ",0h
6 A dd '20'
7 C dd '50'
8 section .bss
9 max resb 10
10 B resb 10
11
12 section .text
13 global _start
14 _start:
15
16 mov eax,msg1
17 call sprint
18
19 mov ecx,B
20 mov edx,10
21 call sread
22
23 mov eax,B
24 call atoi ; Вызов подпрограммы перевода символа в число
25 mov [B],eax ; запись преобразованного числа в 'B'
26
27 mov ecx,[A] ; 'ecx = A'
28 mov [max],ecx ; 'max = A'
29
```

Рисунок 0.14 (текст программы)

```

29
30 cmp ecx,[C] ; Сравниваем 'A' и 'C'
31 jg check_B ; если 'A>C', то переход на метку 'check_B',
32 mov ecx,[C] ; иначе 'ecx = C'
33 mov [max],ecx ; 'max = C'
34
35 check_B:
36 mov eax,max
37 call atoi ; Вызов подпрограммы перевода символа в число
38 mov [max],eax ; запись преобразованного числа в 'max'
39
40 mov ecx,[max]
41 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
42 jg fin ; если 'max(A,C)>B', то переход на 'fin',
43 mov ecx,[B] ; иначе 'ecx = B'
44 mov [max],ecx
45
46 fin:
47 mov eax, msg2
48 call sprintf ; Вывод сообщения 'Наибольшее число: '
49 mov eax,[max]
50 call iprintLF ; Вывод 'max(A,B,C)'
51 call quit ; Выход

```

Рисунок 0.15 (текст программы)

Создадим исполняемый файл и проверим его работу для разных значений B (рисунок 0.16-0.18)

```

dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ gedit lab7-2.asm
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 25
Наибольшее число: 50
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $

```

Рисунок 0.16 (создание исполняемого файла и проверка работы)

```

dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 70
Наибольшее число: 70
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $

```

Рисунок 0.17(создание исполняемого файла и проверка работы)

```

dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 10
Наибольшее число: 50
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $

```

Рисунок 0.18(создание исполняемого файла и проверка работы)

## Изучение структуры файлы листинга

Создадим файл листинга для программы из файла lab7-2.asm (рисунок 0.19-

0.21)

```
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.19 (создание файла листинга)

```
dvkrihloveckiyj@dk6n10 ~ $ mcedit lab7-2.lst
```

Рисунок 0.20 (открытие листинга)

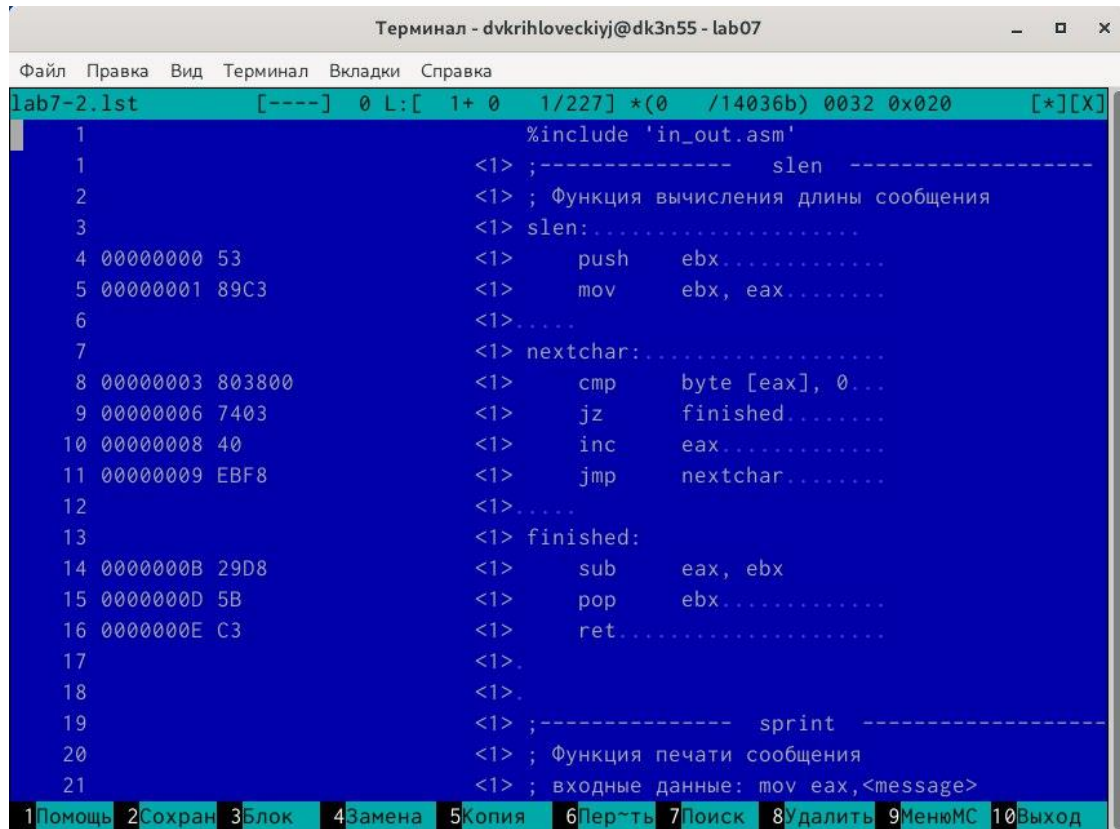


Рисунок 0.21 (открытие листинга)

### Объяснение строк листинга:

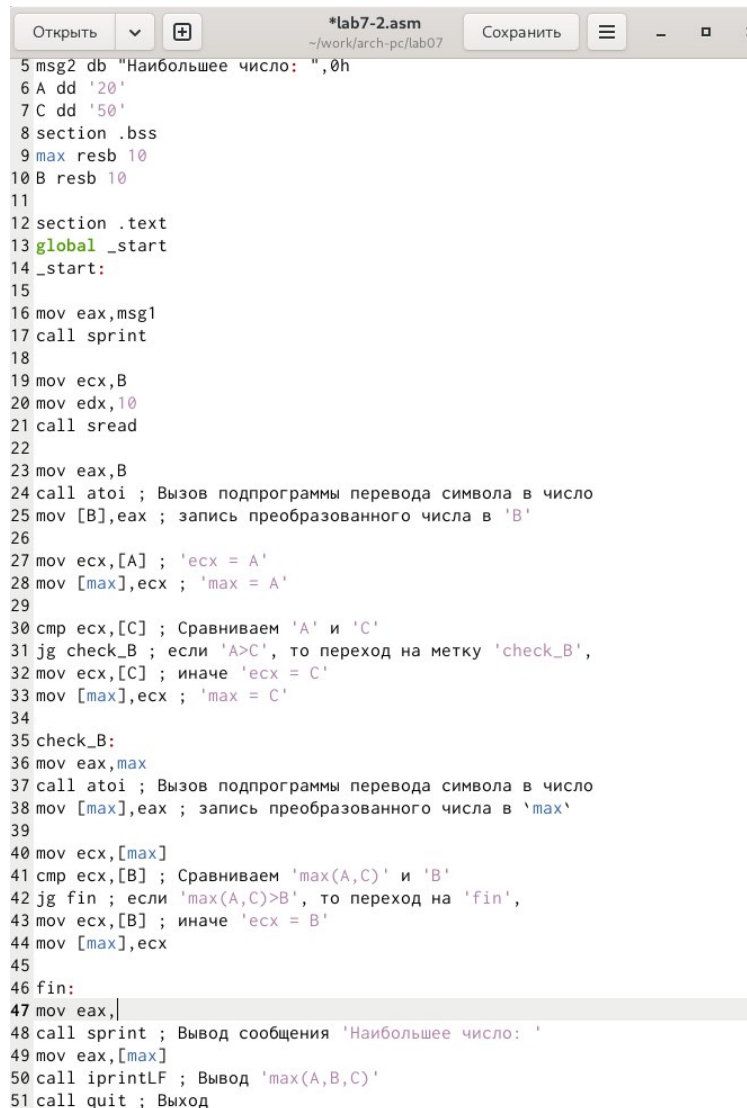
Самое первое значение в файле листинга – номер строки—это номер строки файла листинга (нужно помнить, что номер строки в файле листинга может не соответствовать номеру строки в файле с исходным текстом программы)

Второе вхождение- адрес—это смещение машинного кода от начала текущего сегмента

Далее идут машинный код, представляющий собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, и исходный текст программы — это просто строка исходной программы вместе с комментариями

Откроем файл с программой lab7-2.asm и в любой инструкции с двумя операндами

удалим один операнд. Выполним трансляцию с получением файла листинга (рисунок 0.22-0.24)

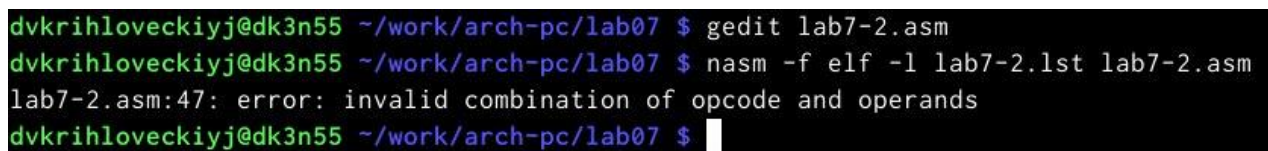


```
*lab7-2.asm
~/work/arch-pc/lab07
Сохранить

5 msg2 db "Наибольшее число: ",0h
6 A dd '20'
7 C dd '50'
8 section .bss
9 max resb 10
10 B resb 10
11
12 section .text
13 global _start
14 _start:
15
16 mov eax,msg1
17 call sprint
18
19 mov ecx,B
20 mov edx,10
21 call sread
22
23 mov eax,B
24 call atoi ; Вызов подпрограммы перевода символа в число
25 mov [B],eax ; запись преобразованного числа в 'B'
26
27 mov ecx,[A] ; 'ecx = A'
28 mov [max],ecx ; 'max = A'
29
30 cmp ecx,[C] ; Сравниваем 'A' и 'C'
31 jg check_B ; если 'A>C', то переход на метку 'check_B',
32 mov ecx,[C] ; иначе 'ecx = C'
33 mov [max],ecx ; 'max = C'
34
35 check_B:
36 mov eax,max
37 call atoi ; Вызов подпрограммы перевода символа в число
38 mov [max],eax ; запись преобразованного числа в 'max'
39
40 mov ecx,[max]
41 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
42 jg fin ; если 'max(A,C)>B', то переход на 'fin',
43 mov ecx,[B] ; иначе 'ecx = B'
44 mov [max],ecx
45
46 fin:
47 mov eax,|
48 call sprint ; Вывод сообщения 'Наибольшее число: '
49 mov eax,[max]
50 call iprintLF ; Вывод 'max(A,B,C)'
51 call quit ; Выход
```

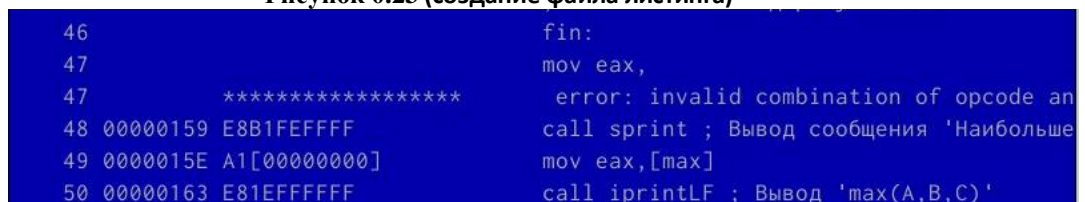
Рисунок 0.22 (текст программы с удалением операнды)

В 47 строке удалим один операнд



```
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ gedit lab7-2.asm
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:47: error: invalid combination of opcode and operands
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.23 (создание файла листинга)



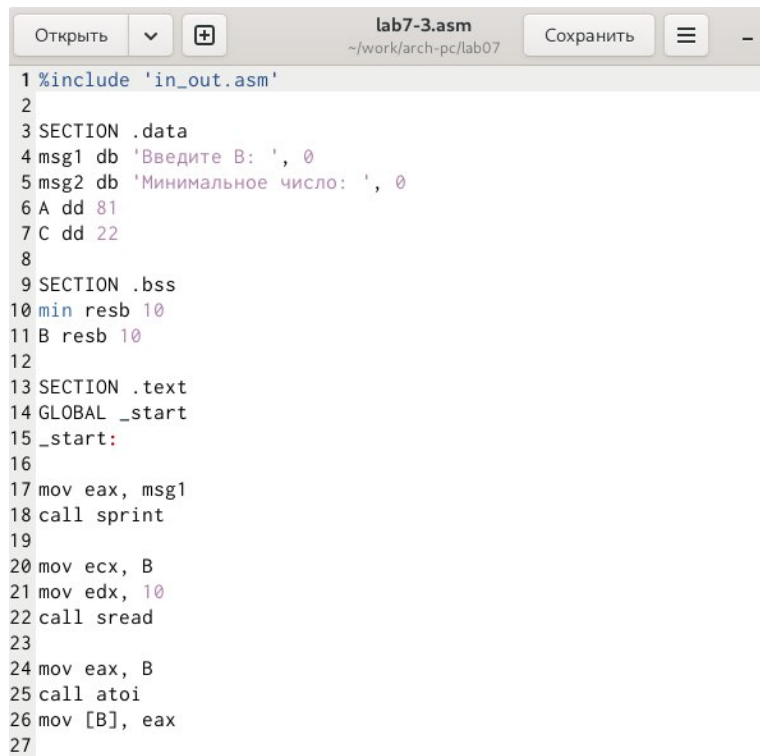
```
46 fin:
47 mov eax,
47 *****
48 00000159 E8B1FEFFFF call sprint ; Вывод сообщения 'Наибольшее
49 0000015E A1[00000000] mov eax,[max]
50 00000163 E81EFFFF call iprintLF ; Вывод 'max(A,B,C)'
```

Рисунок 0.24 (открытие листинга)

В новом файле листинга показывает ошибку, возникшую при попытке трансляции файла. При этом никакие выходные файлы, помимо файла листинга, не появляются

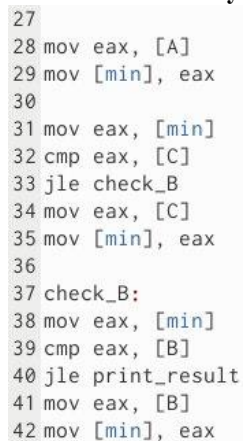
**Задание для самостоятельной работы**

Напишем программу нахождения наименьшей из 3 целочисленных переменных `A`, `B` и `C`. Значения переменных 81, 22, 72 (рисунок 0.25-0.27)



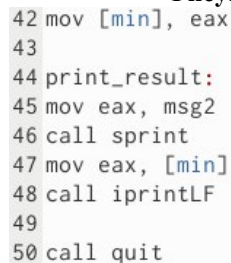
```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1 db 'Введите B: ', 0
5 msg2 db 'Минимальное число: ', 0
6 A dd 81
7 C dd 22
8
9 SECTION .bss
10 min resb 10
11 B resb 10
12
13 SECTION .text
14 GLOBAL _start
15 _start:
16
17 mov eax, msg1
18 call sprint
19
20 mov ecx, B
21 mov edx, 10
22 call sread
23
24 mov eax, B
25 call atoi
26 mov [B], eax
27
```

Рисунок 0.25 (текст программы)



```
27
28 mov eax, [A]
29 mov [min], eax
30
31 mov eax, [min]
32 cmp eax, [C]
33 jle check_B
34 mov eax, [C]
35 mov [min], eax
36
37 check_B:
38 mov eax, [min]
39 cmp eax, [B]
40 jle print_result
41 mov eax, [B]
42 mov [min], eax
```

Рисунок 0.26 (текст программы)



```
42 mov [min], eax
43
44 print_result:
45 mov eax, msg2
46 call sprint
47 mov eax, [min]
48 call iprintLF
49
50 call quit
```

Рисунок 0.27 (текст программы)

Создайте исполняемый файл и проверьте его работу (рисунок 0.28)

```
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ gedit lab7-3.asm
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-3.lst lab7-3.asm
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-3
Введите B: 22
Минимальное число: 22
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-3
Введите B: 72
Минимальное число: 22
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.28 (создание исполняемого файла и проверка его работы)

Напишите программу, которая для введенных с клавиатуры значений  $x$  и  $a$  вычисляет значение заданной функции  $f(x, a)$  и выводит результат вычислений. Вид функции  $f(x, a) = [ (3a+1, \text{при } x < a), (3x+1, \text{при } x \geq a) ]$  (рисунок 0.29-0.30)

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg_x: DB 'Введите значение переменной x: ', 0
4 msg_a: DB 'Введите значение переменной a: ', 0
5 res: DB 'Результат: ', 0
6 SECTION .bss
7 x: RESB 80
8 a: RESB 80
9 SECTION .text
10 GLOBAL _start
11 _start:
12     mov eax, msg_x
13     call sprint
14     mov ecx, x
15     mov edx, 80
16     call sread
17     mov eax, x
18     call atoi
19     mov edi, eax
20
21     mov eax, msg_a
22     call sprint
23     mov ecx, a
24     mov edx, 80
25     call sread
26     mov eax, a
27     call atoi
28     mov esi, eax
29
30     cmp edi, esi
31     jl less_case
```

Рисунок 0.29 (текст программы)

```

32
33     mov eax, edi
34     mov ebx, 3
35     mul ebx
36     add eax, 1
37     jmp print_result
38
39 less_case:
40     mov eax, esi
41     mov ebx, 3
42     mul ebx
43     add eax, 1
44
45 print_result:
46     mov edi, eax
47     mov eax, res
48     call sprint
49     mov eax, edi
50     call iprintLF
51     call quit

```

Рисунок 0.30 (текст программы)

Создадим исполняемый файл и проверим его работу для значений  $\square$  и  $\square$   
(рисунок 0.31-0.35)

```

dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ touch lab7-4.asm
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ gedit lab7-4.asm
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm

```

Рисунок 0.31 (создание исполняемого файла)

```

dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o

```

Рисунок 0.32 (создание исполняемого файла)

```

dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-4

```

Рисунок 0.33 (запуск программы)

```

dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-4
Введите значение переменной x: 2
Введите значение переменной a: 3
Результат: 10

```

Рисунок 0.34 (проверка результатов)

```

dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-4
Введите значение переменной x: 4
Введите значение переменной a: 2
Результат: 13
dvkrihloveckiyj@dk3n55 ~/work/arch-pc/lab07 $

```

Рисунок 0.35 (проверка результатов)

## **5 Выводы**

Мы изучили команды условного и безусловного перехода, приобрели навыки написания программ с использованием переходов, ознакомились с назначением и структурой файла листинга.

## Список литературы

1. GDB: The GNU Project Debugger.—URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual.—2016.—URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center.—2021.—URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials.—2021.—URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658.—URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference.—O'Reilly Media, 2016.—156 с.—ISBN 978-1491941591.
7. The NASM documentation.—2021.—URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash.—Packt Publishing, 2017.—502 с.—ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ.—М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER.—М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ систем.—М.: Юрайт, 2016.
12. Расширенный ассемблер: NASM.—2021.—URL: <https://www.opennet.ru/docs/RUS/nasm/>.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX.—2-е изд.—БХВ Петербург, 2010.—656 с.—ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix.—2-е изд.—М. : МАКС Пресс, 2011.—URL: [http://www.stolyarov.info/books/asm\\_unix](http://www.stolyarov.info/books/asm_unix).
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы.—4-е изд.—СПб.: Питер, 2015. —1120 с.—(Классика Computer Science)