

# **Отчёт по лабораторной работе №8**

**Команды условного и безусловного переходов в NASM.  
Программирование ветвлений**

Аскеров Александр Эдуардович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>5</b>
2.1	Реализация переходов в NASM . . . . .	5
2.2	Изучение структуры файлов листинга . . . . .	13
2.3	Задание для самостоятельной работы . . . . .	17
<b>3</b>	<b>Выводы</b>	<b>19</b>

## Список иллюстраций

2.1	Создание каталога и файла lab8-1.asm . . . . .	5
2.2	Программа из листинга 8.1 . . . . .	6
2.3	Результат работы программы lab8-1 . . . . .	6
2.4	Изменённый текст программы lab8-1 (листинг 8.2) . . . . .	8
2.5	Результат работы программы lab8-1 . . . . .	9
2.6	Изменённый текст программы lab8-1 . . . . .	10
2.7	Результат работы программы lab8-1 . . . . .	11
2.8	Создание файла lab8-2.asm . . . . .	11
2.9	Программа из листинга 8.3 (фрагмент) . . . . .	12
2.10	Результат работы программы lab8-2 . . . . .	13
2.11	Создание файла листинга . . . . .	13
2.12	Содержимое листинга lab8-2.lst . . . . .	14
2.13	Фрагмент листинга lab8-2.lst . . . . .	15
2.14	Удалим операнд, выделенный красным . . . . .	16
2.15	Выполнение трансляции с получением файла листинга . . . . .	16
2.16	Сообщение об ошибке в листинге . . . . .	17
2.17	Результат работы программы lab8-3.asm (вариант 19) . . . . .	17
2.18	Результат работы программы lab8-4.asm (вариант 19) . . . . .	18

# 1 Цель работы

Изучить команды условного и безусловного переходов. Приобрести навыки написания программ с использованием переходов. Познакомиться с назначением и структурой файла листинга.

## 2 Выполнение лабораторной работы

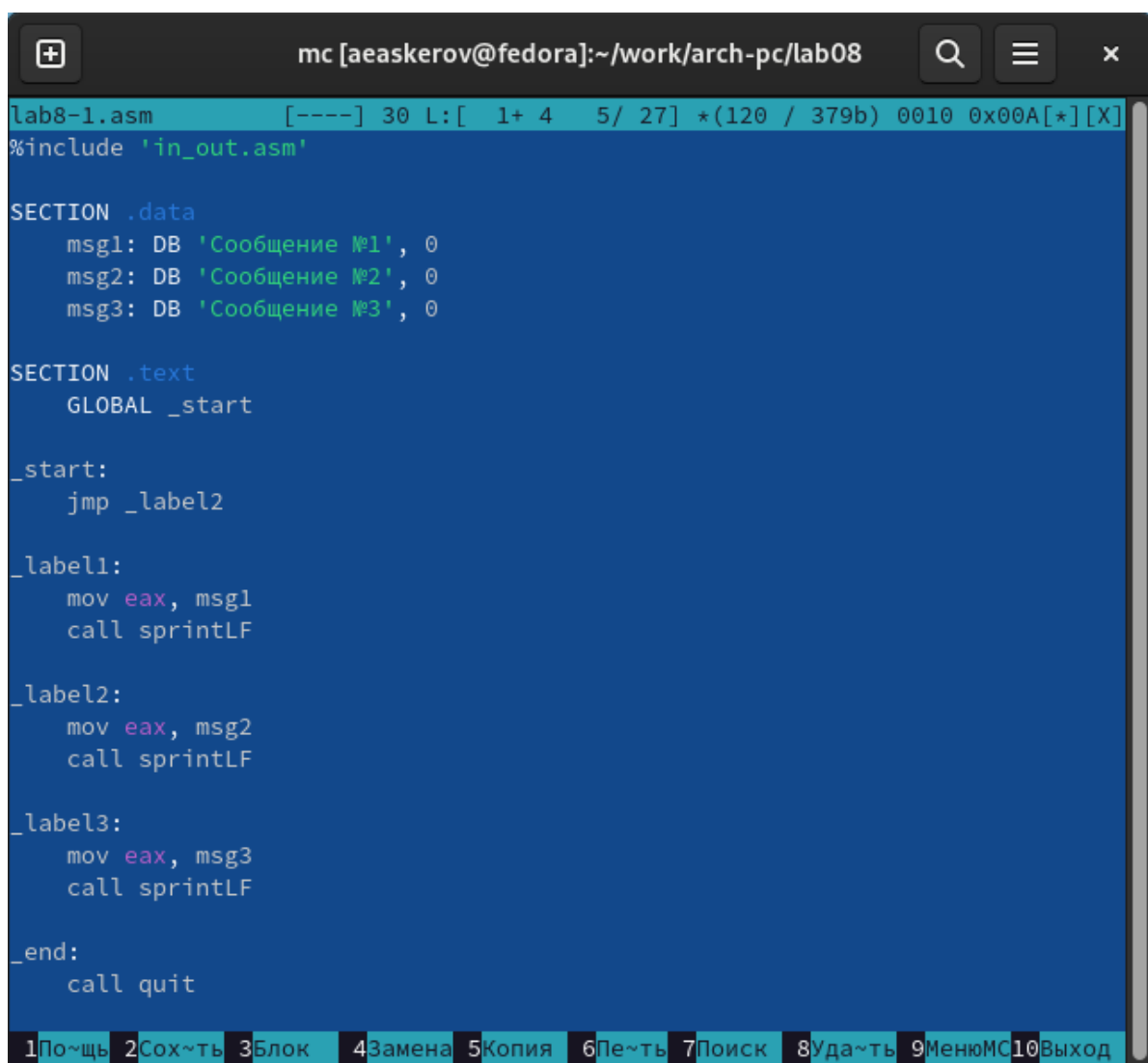
### 2.1 Реализация переходов в NASM

1. Создадим каталог для программ лабораторной работы №8, перейдём в него и создадим файл lab8-1.asm.

```
[aeaskerov@fedora ~]$ mkdir ~/work/arch-pc/lab08  
[aeaskerov@fedora ~]$ cd ~/work/arch-pc/lab08  
[aeaskerov@fedora lab08]$ touch lab8-1.asm  
[aeaskerov@fedora lab08]$
```

Рис. 2.1: Создание каталога и файла lab8-1.asm

2. Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Введём в файл lab8-1.asm текст программы из листинга 8.1.



```
lab8-1.asm [----] 30 L:[ 1+ 4 5/ 27] *(120 / 379b) 0010 0x00A[*][X]
#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

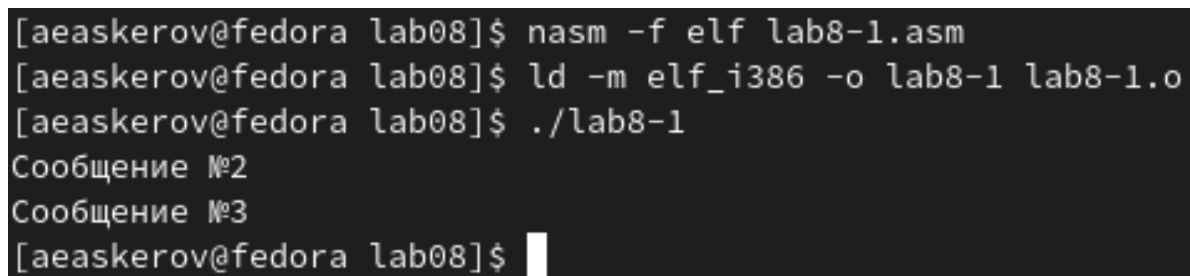
_label3:
    mov eax, msg3
    call sprintLF

_end:
    call quit
```

1По~щ~ь 2Сох~ть 3Блок 4Замена 5Копия 6Пе~ть 7Поиск 8Уда~ть 9МенюМС10Выход

Рис. 2.2: Программа из листинга 8.1

Создадим исполняемый файл и запустим его. Результат работы данной программы будет следующим.

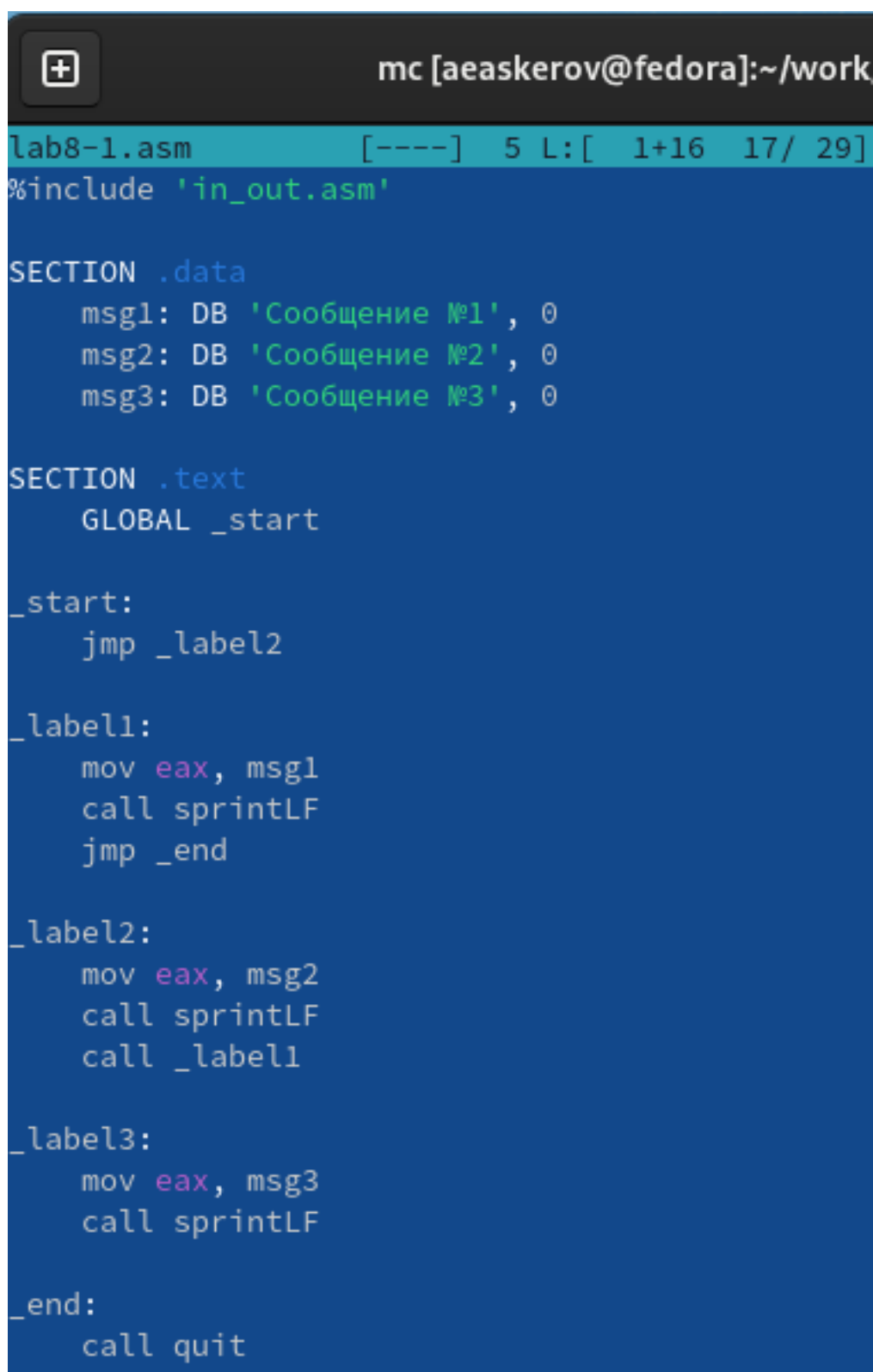


```
[aeaskerov@fedora lab08]$ nasm -f elf lab8-1.asm
[aeaskerov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[aeaskerov@fedora lab08]$ ./lab8-1
Сообщение №2
Сообщение №3
[aeaskerov@fedora lab08]$
```

Рис. 2.3: Результат работы программы lab8-1

Таким образом, 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 вперёд, 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 8.2.



```
lab8-1.asm [----] 5 L:[ 1+16 17/ 29]
#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
    jmp _end

_label2:
    mov eax, msg2
    call sprintf
    call _label1

_label3:
    mov eax, msg3
    call sprintf

_end:
    call quit
```

Рис. 2.4: Изменённый текст программы lab8-1 (листинг 8.2)

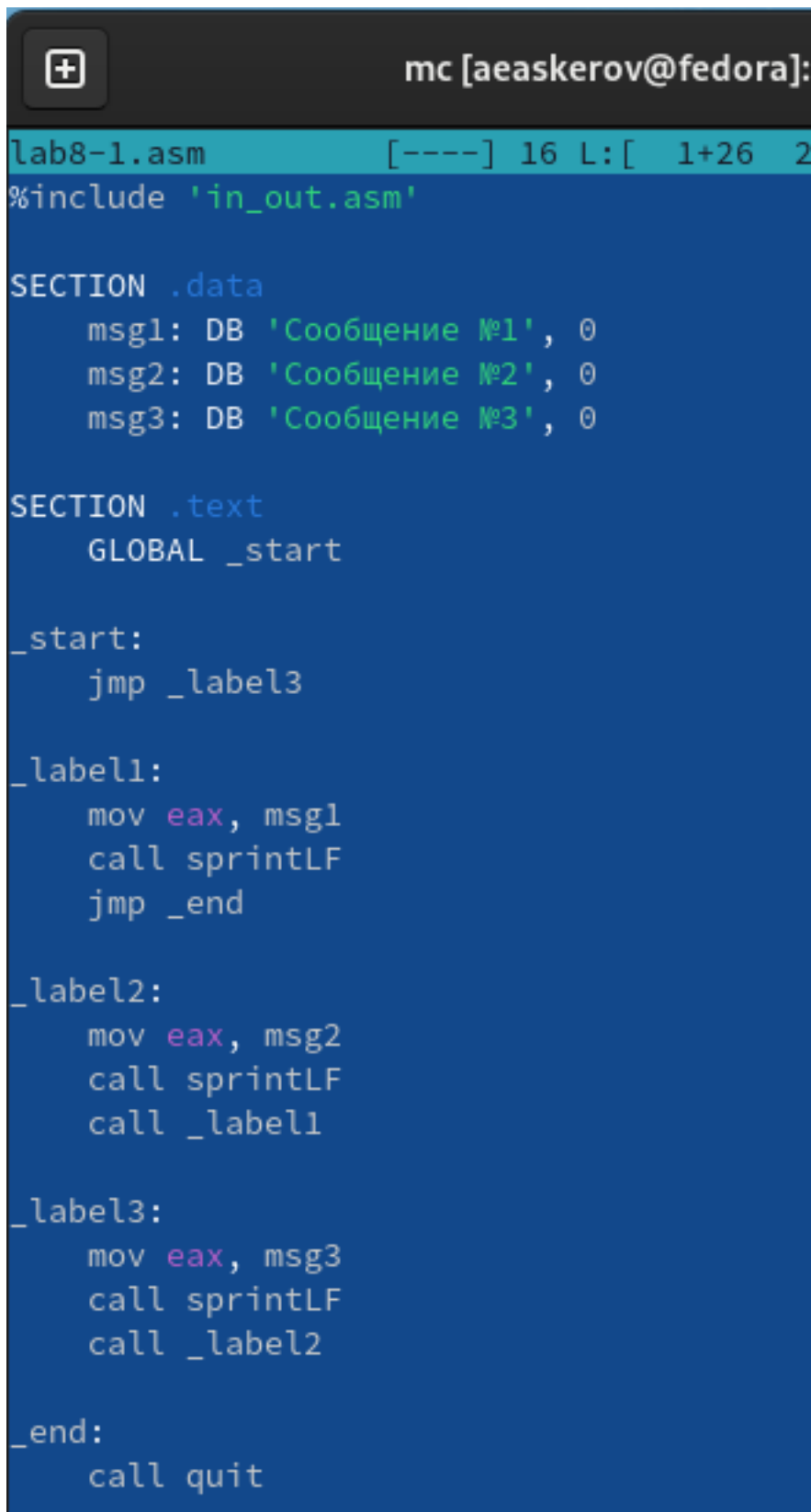


Создадим исполняемый файл и проверим его работу.

```
[aeaskerov@fedora lab08]$ nasm -f elf lab8-1.asm
[aeaskerov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[aeaskerov@fedora lab08]$ ./lab8-1
Сообщение №2
Сообщение №1
[aeaskerov@fedora lab08]$
```

Рис. 2.5: Результат работы программы lab8-1

Изменим текст программы, добавив или изменив инструкции `jmp` так, чтобы вывод программы был следующим: сначала Сообщение №3, потом Сообщение №2 и потом Сообщение №1.



```
lab8-1.asm      [----] 16 L:[ 1+26 2
%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
    call _label1

_label3:
    mov eax, msg3
    call sprintLF
    call _label2

_end:
    call quit
```

Рис. 2.6: Изменённый текст программы lab8-1

```
[aeaskerov@fedora lab08]$ nasm -f elf lab8-1.asm
[aeaskerov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[aeaskerov@fedora lab08]$ ./lab8-1
Сообщение №3
Сообщение №2
Сообщение №1
[aeaskerov@fedora lab08]$
```


Рис. 2.7: Результат работы программы lab8-1

3. Использование инструкции `jmp` приводит к переходу в любом случае. Однако часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить, если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры.

Создадим файл `lab8-2.asm` в каталоге `~/work/arch-pc/lab08`. Внимательно изучим текст программы из листинга 8.3 и введём в `lab8-2.asm`.

```
[aeaskerov@fedora lab08]$ touch lab8-2.asm
[aeaskerov@fedora lab08]$
```

Рис. 2.8: Создание файла lab8-2.asm



```
lab8-2.asm [----] 7 L:[ 1+14
#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]
    mov [max],ecx

    cmp ecx,[C]
    jg check_B
```

1Поиск 2Сохранить 3Блок 4Замена 5Копия

Рис. 2.9: Программа из листинга 8.3 (фрагмент)

Создадим исполняемый файл и проверим его работу для разных значений В.

```
[aeaskerov@fedora lab08]$ nasm -f elf lab8-2.asm
[aeaskerov@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[aeaskerov@fedora lab08]$ ./lab8-2
Введите В: 45
Наибольшее число: 50
[aeaskerov@fedora lab08]$ ./lab8-2
Введите В: 65
Наибольшее число: 65
[aeaskerov@fedora lab08]$
```

Рис. 2.10: Результат работы программы lab8-2

Обратим внимание, в данном примере переменные А и С сравниваются как символы, а переменная В и максимум из А и С как числа (для этого используется функция `atoi` преобразования символа в число). Это сделано для демонстрации того, как сравниваются данные. Данную программу можно упростить и сравнивать все 3 переменные как символы (т.е. не использовать функцию `atoi`). Однако если переменные преобразовать из символов в числа, над ними можно корректно проводить арифметические операции.

## 2.2 Изучение структуры файлов листинга

4. 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 `lab8-2.asm`.

```
[aeaskerov@fedora lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
[aeaskerov@fedora lab08]$
```

Рис. 2.11: Создание файла листинга

Откроем файл листинга lab8-2.lst с помощью любого текстового редактора, например, mcedit.

```
lab8-2.lst [----] 0 L: [ 1+ 0 1/227] *(0 /13451b) 0032 0x020 [*] [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>
23      <1> sprint:
24 0000000F 52      <1>      push    edx
25 00000010 51      <1>      push    ecx
26 00000011 53      <1>      push    ebx
27 00000012 50      <1>      push    eax
28 00000013 E8E8FFFFFF <1>      call    slen
29      <1>.....
30 00000018 89C2    <1>      mov     edx, eax
31 0000001A 58      <1>      pop     eax
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9МенюМС 10Выход

Рис. 2.12: Содержимое листинга lab8-2.lst

Внимательно ознакомимся с его форматом и содержанием. Подробно объясним содержимое трёх строк файла листинга по выбору.

```

35                                     check_B:
36 00000130 B8[00000000]             <----->mov eax,max
37 00000135 E862FFFFFF             <----->call atoi
38 0000013A A3[00000000]             <----->mov [max],eax

```

Рис. 2.13: Фрагмент листинга lab8-2.lst

36, 37, 38 – номера строк.

00000130, 00000135, 0000013A – адрес строки.

B8[00000000], E862FFFFFF, A3[00000000] – машинный код.

“mov eax,max”, “call atoi”, “mov [max],eax” – исходный текст программы.

Откроем файл с программой lab8-2.asm и в любой инструкции с двумя операндами удалим один операнд. Выполним трансляцию с получением файла листинга.

```
mc [aeaskerov@fedora]:  
lab8-2.asm [----] 0  
    mov eax,B  
    call atoi  
    mov [B],eax  
  
    mov ecx,[A]  
    mov [max],ecx  
  
    cmp ecx,[C]  
    jg check_B  
    mov ecx,[C]  
    mov [max],ecx  
  
    check_B:  
<----->mov eax,max  
<----->call atoi  
<----->mov [max],eax  
  
<----->mov ecx,[max]  
<----->cmp ecx,[B]  
<----->jg fin  
<----->mov ecx,[B]  
<----->mov [max],ecx  
1По~щъ 2Со~ть 3Блок 4За~на
```

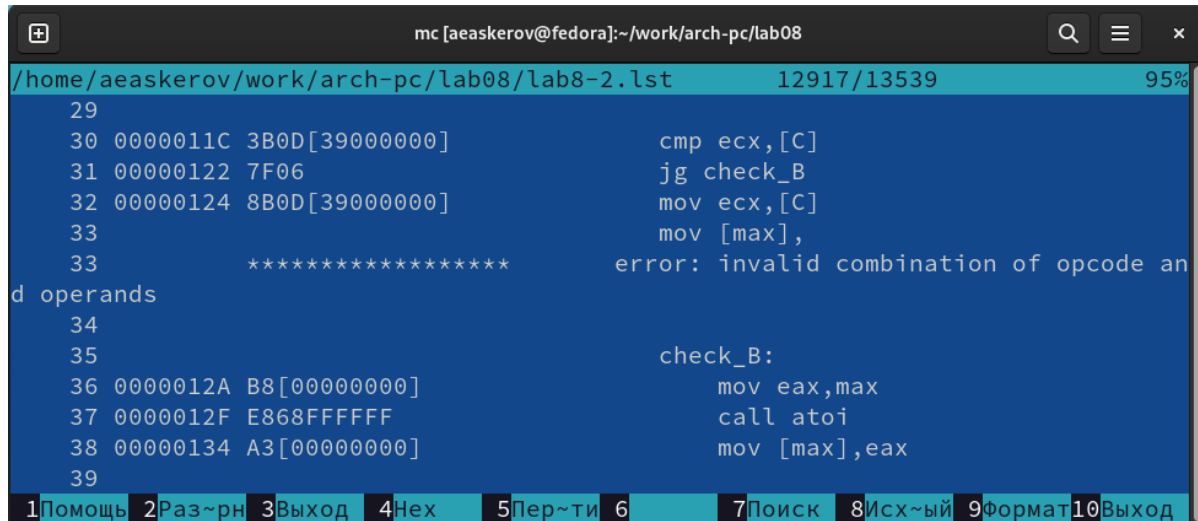
Рис. 2.14: Удалим операнд, выделенный красным

```
[aeaskerov@fedora lab08]$ nasm -f elf -l lab8-2.lst lab8-2.  
asm  
lab8-2.asm:30: error: invalid combination of opcode and ope  
rands  
[aeaskerov@fedora lab08]$
```

Рис. 2.15: Выполнение трансляции с получением файла листинга



Мы видим, что в результате удаления операнда вышла ошибка и никакие файлы не сформировались, а в листинге появилось сообщение об ошибке.

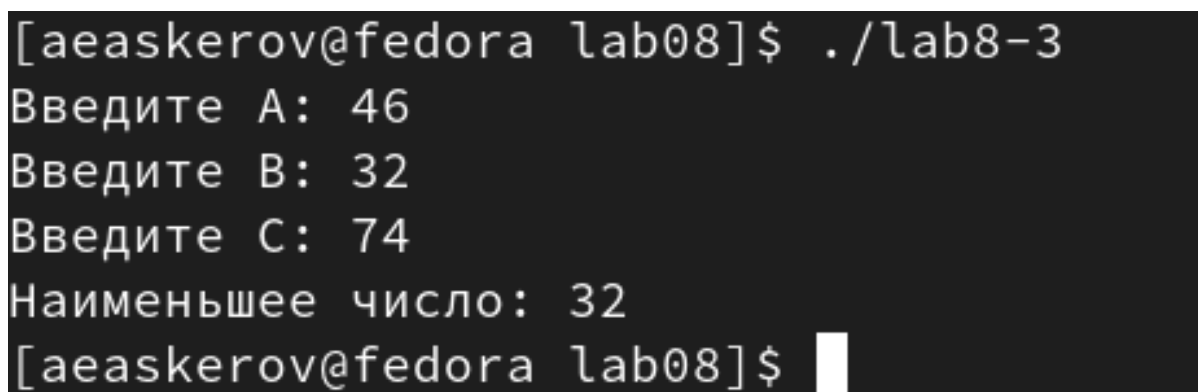


```
mc [aeaskerov@fedora]:~/work/arch-pc/lab08
/home/aeaskerov/work/arch-pc/lab08/lab8-2.lst 12917/13539 95%
29
30 0000011C 3B0D[39000000]      cmp ecx,[C]
31 00000122 7F06                      jg  check_B
32 00000124 8B0D[39000000]      mov ecx,[C]
33                      mov [max],
33                      ***** error: invalid combination of opcode and operands
34
35                      check_B:
36 0000012A B8[00000000]      mov eax,max
37 0000012F E868FFFFFF      call atoi
38 00000134 A3[00000000]      mov [max],eax
39
1Помощь 2Раз~рн 3Выход 4Нех 5Пер~ти 6 7Поиск 8Исх~ый 9Формат10Выход
```

Рис. 2.16: Сообщение об ошибке в листинге

## 2.3 Задание для самостоятельной работы

1. Напишем программу нахождения наименьшей из 3 целочисленных переменных а, b и с. Значения переменных выберем из табл. 8.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создадим исполняемый файл и проверим его работу.



```
[aeaskerov@fedora lab08]$ ./lab8-3
Введите A: 46
Введите B: 32
Введите C: 74
Наименьшее число: 32
[aeaskerov@fedora lab08]$
```

Рис. 2.17: Результат работы программы lab8-3.asm (вариант 19)

2. Напишем программу, которая для введенных с клавиатуры значений  $x$  и  $a$  вычисляет значение заданной функции  $f(x)$  и выводит результат вычислений. Вид функции  $f(x)$  выберем из таблицы 8.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создадим исполняемый файл и проверим его работу для значений  $x$  и  $a$  из 8.6.

```
[aeaskerov@fedora lab08]$ ./lab8-4
Введите x: 4
Введите a: 5
Результат: 4
[aeaskerov@fedora lab08]$ ./lab8-4
Введите x: 3
Введите a: 2
Результат: 5
[aeaskerov@fedora lab08]$
```

Рис. 2.18: Результат работы программы lab8-4.asm (вариант 19)

## **3 Выводы**

Изучены команды условного и безусловного переходов. Приобретены навыки написания программ с использованием переходов. Ознакомлен с назначением и структурой файла листинга.