

# **Отчёт по лабораторной работе 7**

**Дисциплина: архитектура компьютера**

Маныев Ресулбег Алексеевич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Реализация переходов в NASM . . . . .	6
2.2	Изучение структуры файлы листинга . . . . .	14
2.3	Задание для самостоятельной работы . . . . .	17
<b>3</b>	<b>Выводы</b>	<b>22</b>

## Список иллюстраций

2.1	Код программы lab7-1.asm . . . . .	7
2.2	Компиляция и запуск программы lab7-1.asm . . . . .	8
2.3	Код программы lab7-1.asm . . . . .	9
2.4	Компиляция и запуск программы lab7-1.asm . . . . .	10
2.5	Код программы lab7-1.asm . . . . .	11
2.6	Компиляция и запуск программы lab7-1.asm . . . . .	12
2.7	Код программы lab7-2.asm . . . . .	13
2.8	Компиляция и запуск программы lab7-2.asm . . . . .	14
2.9	Файл листинга lab7-2 . . . . .	15
2.10	Ошибка трансляции lab7-2 . . . . .	16
2.11	Файл листинга с ошибкой lab7-2 . . . . .	17
2.12	Код программы prog-1.asm . . . . .	18
2.13	Компиляция и запуск программы prog-1.asm . . . . .	19
2.14	Код программы prog-2.asm . . . . .	20
2.15	Компиляция и запуск программы prog-2.asm . . . . .	21

## Список таблиц

# 1 Цель работы

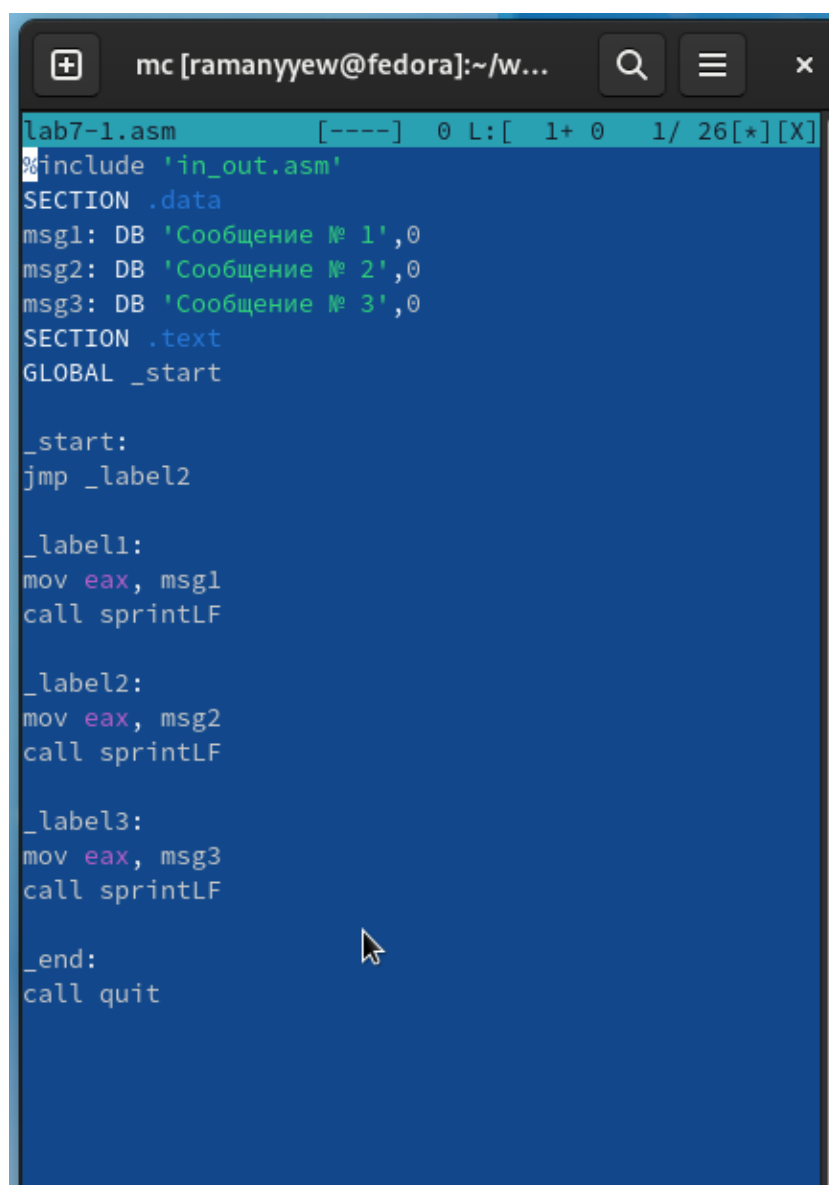
Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

## 2 Выполнение лабораторной работы

### 2.1 Реализация переходов в NASM

Я создал каталог для программ лабораторной работы № 7 и файл lab7-1.asm.

Инструкция `jmp` в NASM используется для выполнения безусловных переходов. Рассмотрим пример программы, в которой используется инструкция `jmp`. Написал текст программы из листинга 7.1 в файле lab7-1.asm. (рис. [2.1])



```
lab7-1.asm [----] 0 L: [ 1+ 0 1/ 26[*] [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

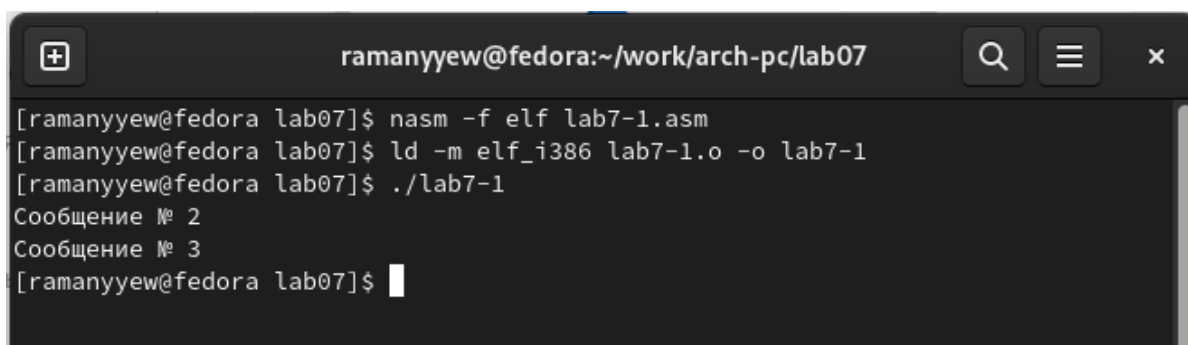
_label2:
mov eax, msg2
call sprintLF

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.1: Код программы lab7-1.asm

Создал исполняемый файл и запустил его. (рис. [2.2])

A terminal window with a dark background. The title bar shows the user 'ramanyyew@fedora' and the directory '~/work/arch-pc/lab07'. The terminal contains the following text:

```
[ramanyyew@fedora lab07]$ nasm -f elf lab7-1.asm
[ramanyyew@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[ramanyyew@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[ramanyyew@fedora lab07]$
```

Рис. 2.2: Компиляция и запуск программы lab7-1.asm

Инструкция `jmp` позволяет осуществлять переходы не только вперед, но и назад. Мы изменим программу так, чтобы она сначала выводила “Сообщение № 2”, затем “Сообщение № 1” и завершала работу. Для этого мы добавим в текст программы после вывода “Сообщения № 2” инструкцию `jmp` с меткой `_label1` (чтобы перейти к инструкциям вывода “Сообщения № 1”) и после вывода “Сообщения № 1” добавим инструкцию `jmp` с меткой `_end` (чтобы перейти к инструкции `call quit`).

Изменил текст программы в соответствии с листингом 7.2. (рис. [2.3] [2.4])





```
lab7-1.asm [----] 9 L: [ 1+26 27/ 28[*] [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
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.3: Код программы lab7-1.asm

```
Сообщение № 3
[ramanyu@fedora lab07]$ nasm -f elf lab7-1.asm
[ramanyu@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[ramanyu@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 1
[ramanyu@fedora lab07]$
```

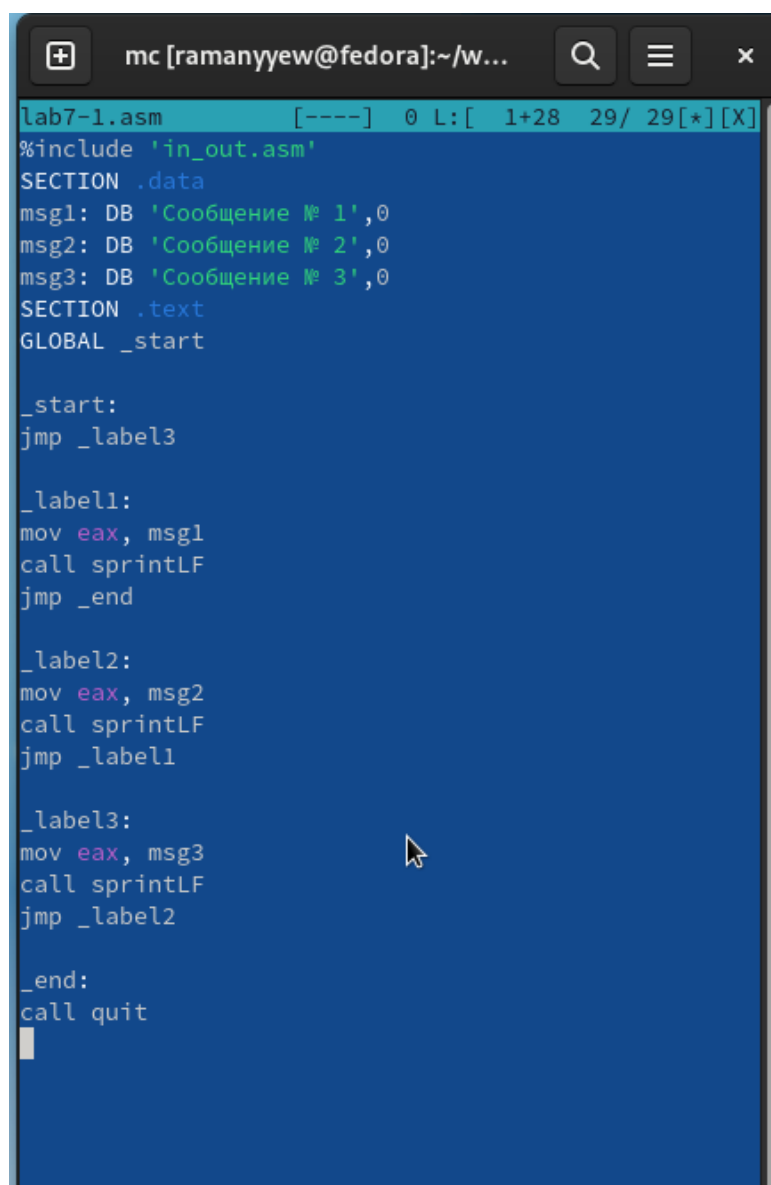
Рис. 2.4: Компиляция и запуск программы lab7-1.asm

Изменил текст программы (рис. [2.5] [2.6]), изменив инструкции `jmp`, чтобы вывод программы был следующим:

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
lab7-1.asm [----] 0 L: [ 1+28 29/ 29[*] [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 _label3

_label1:
mov eax, msg1
call sprintf
jmp _end

_label2:
mov eax, msg2
call sprintf
jmp _label1

_label3:
mov eax, msg3
call sprintf
jmp _label2

_end:
call quit
```

Рис. 2.5: Код программы lab7-1.asm

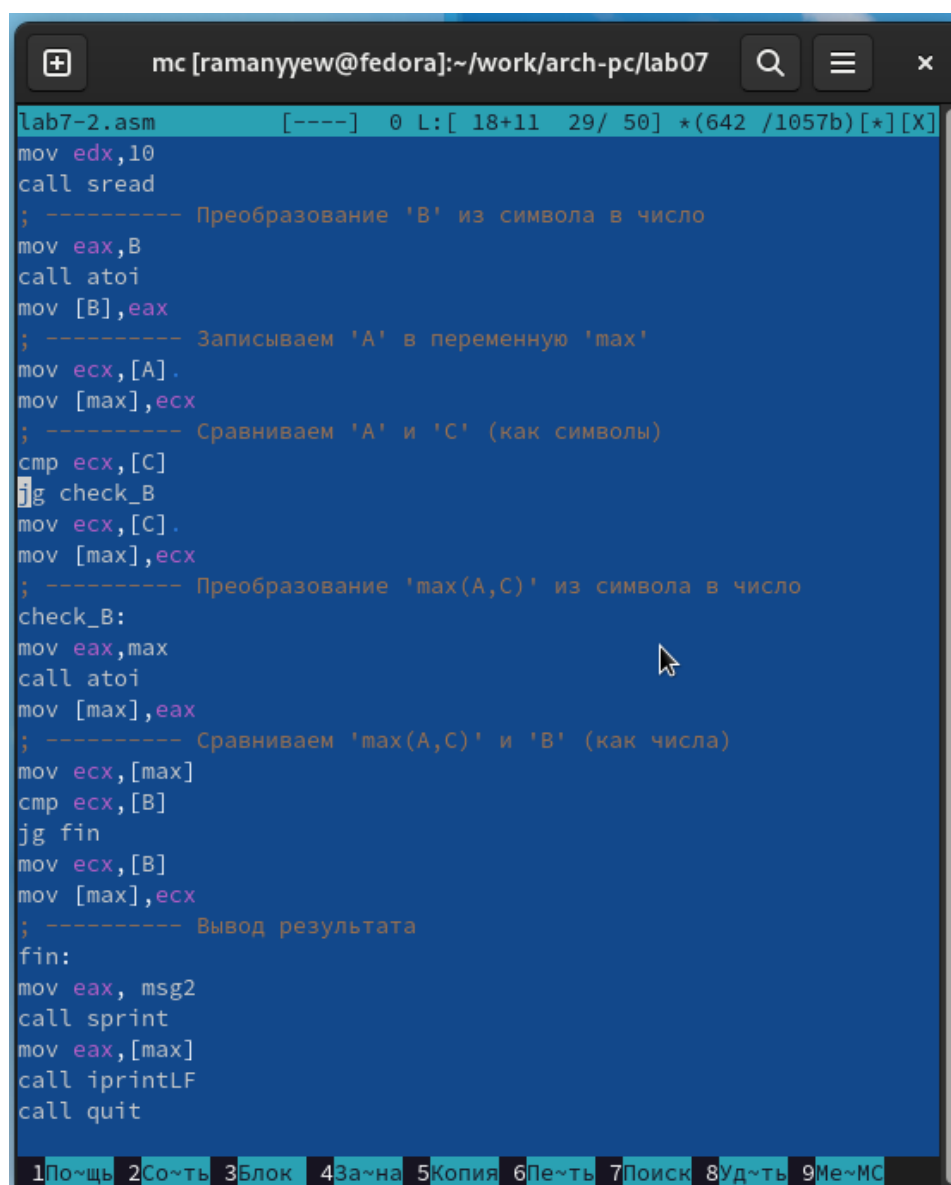
```
[ramanyyew@fedora lab07]$  
[ramanyyew@fedora lab07]$ nasm -f elf lab7-1.asm  
[ramanyyew@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1  
[ramanyyew@fedora lab07]$ ./lab7-1  
Сообщение № 3  
Сообщение № 2  
Сообщение № 1  
[ramanyyew@fedora lab07]$
```

Рис. 2.6: Компиляция и запуск программы lab7-1.asm

Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, то есть переход должен происходить, если выполнено какое-либо условие.

Давайте рассмотрим программу, которая определяет и выводит на экран наибольшую из трех целочисленных переменных: А, В и С. Значения для А и С задаются в программе, а значение В вводится с клавиатуры.

Создал исполняемый файл и проверил его работу для разных значений В. (рис. [2.7] [2.8])



```
mc [ramanyyew@fedora]:~/work/arch-pc/lab07 [lab7-2.asm] [----] 0 L: [ 18+11 29/ 50] *(642 /1057b) [*] [X]
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A]
mov [max],ecx
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi
mov [max],eax
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx
; ----- Вывод результата
fin:
mov eax,msg2
call sprint
mov eax,[max]
call iprintLF
call quit
```

1Почть 2Сочть 3Блок 4За~на 5Копия 6Петь 7Поиск 8Уд~ть 9Ме~МС

Рис. 2.7: Код программы lab7-2.asm

```
[ramanyu@fedora lab07]$ nasm -f elf lab7-2.asm
[ramanyu@fedora lab07]$ ld -m elf_i386 lab7-2.o -o lab7-2
[ramanyu@fedora lab07]$ ./lab7-2
Введите В: 10
Наибольшее число: 50
[ramanyu@fedora lab07]$ ./lab7-2
Введите В: 20
Наибольшее число: 50
[ramanyu@fedora lab07]$ ./lab7-2 50
Введите В: 50
Наибольшее число: 50
[ramanyu@fedora lab07]$ ./lab7-2
Введите В: 60
Наибольшее число: 60
[ramanyu@fedora lab07]$ ./lab7-2
Введите В: 100
Наибольшее число: 100
[ramanyu@fedora lab07]$
```

Рис. 2.8: Компиляция и запуск программы lab7-2.asm

## 2.2 Изучение структуры файлы листинга

Обычно `nasm` создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ `-l` и задав имя файла листинга в командной строке.

Создал файл листинга для программы из файла `lab7-2.asm` (рис. [2.9])

```
lab7-2.lst      [----]  0 L:[177+14 191/225] *(11649/13771b) 0032 0x020 [X] [X]
4 0000001C BED0BBD18CD188D0B5-      A dd '20'
4 00000025 D0B520D187D0B8D181-      C dd '50'
4 0000002E D0BBD0BE3A2000.....      section .bss
5 00000035 32300000                      max resb 10
6 00000039 35300000                      B resb 10
7                                     section .text
8 00000000 <res Ah>                      global _start
9 0000000A <res Ah>                      _start:
10                                     ; ----- Вывод сообщения 'Введите B: '
11                                     mov eax,msg1
12                                     call sprint
13                                     ; ----- Ввод 'B'
14 000000E8 B8[00000000]                mov ecx,B
15 000000ED E81DFFFFFF                mov edx,10
16                                     call sread
17 000000F2 B9[0A000000]                ; ----- Преобразование 'B' из символа в число
18 000000F7 BA0A000000                mov eax,B
19 000000FC E842FFFFFF                call atoi
20                                     ; ----- Записываем 'A' в переменную 'max'
21 00000101 B8[0A000000]                mov ecx,[A]
22 00000106 E891FFFFFF                mov [max],ecx
23 0000010B A3[0A000000]                ; ----- Сравниваем 'A' и 'C' (как символы)
24                                     cmp ecx,[C]
25 00000110 8B0D[35000000]                jg check_B
26 00000116 890D[00000000]                mov ecx,[C]
27                                     mov [max],ecx
28 0000011C 3B0D[39000000]                ; ----- Преобразование 'max(A,C)' из символа в число
29 00000122 7F0C                      check_B:
30 00000124 8B0D[39000000]                mov eax,max
31 0000012A 890D[00000000]                mov eax,max
32                                     mov eax,max
33                                     mov eax,max
34 00000130 B8[00000000]
```

Рис. 2.9: Файл листинга lab7-2

Внимательно ознакомился с его форматом и содержимым. Подробно объясню содержимое трёх строк файла листинга.

*строка 192*

- 17 - номер строки в подпрограмме
- 000000F2 - адрес
- B9[0A000000] - машинный код
- mov ecx,B - код программы - копирует B в ecx

*строка 193*

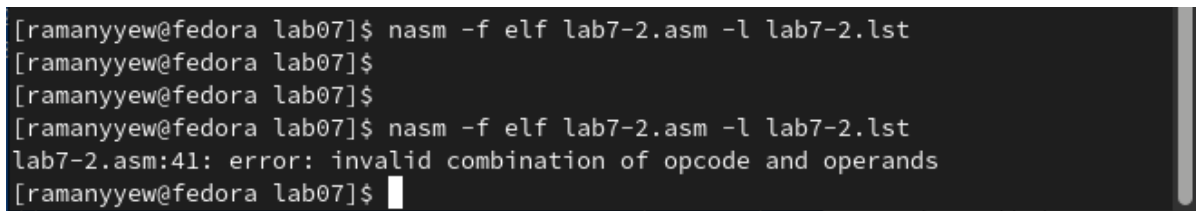
- 18 - номер строки в подпрограмме
- 000000F7 - адрес

- BA0A000000 - машинный код
- mov edx,10 - код программы - копирует 10 в edx

*строка 194*

- 19 - номер строки в подпрограмме
- 000000FC - адрес
- E842FFFFFF - машинный код
- call sread - код программы - вызов подпрограммы чтения

Открыл файл с программой lab7-2.asm и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга. (рис. [2.10]) (рис. [2.11])



```
[ramanyyew@fedora lab07]$ nasm -f elf lab7-2.asm -l lab7-2.lst
[ramanyyew@fedora lab07]$
[ramanyyew@fedora lab07]$
[ramanyyew@fedora lab07]$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:41: error: invalid combination of opcode and operands
[ramanyyew@fedora lab07]$
```

Рис. 2.10: Ошибка трансляции lab7-2



```
mc [ramanyyew@fedora:~/work/arch-pc/lab07]
lab7-2.lst [----] 0 L:[194+32 226/226] *(13859/13859b) <EOF>
19 000000FC E842FFFFFF call sread
20 ; ----- Преобразование 'B' из символа в число
21 00000101 B8[0A000000] mov eax,B
22 00000106 E891FFFFFF call atoi
23 0000010B A3[0A000000] mov [B],eax
24 ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000] mov ecx,[A]
26 00000116 890D[00000000] mov [max],ecx
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000] cmp ecx,[C]
29 00000122 7F0C jg check_B
30 00000124 8B0D[39000000] mov ecx,[C]
31 0000012A 890D[00000000] mov [max],ecx
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 00000130 B8[00000000] mov eax,max
35 00000135 E862FFFFFF call atoi
36 0000013A A3[00000000] mov [max],eax
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013F 8B0D[00000000] mov ecx,[max]
39 00000145 3B0D[0A000000] cmp ecx,[B]
40 0000014B 7F06 jg fin
41 mov ecx,
41 *****
42 0000014D 890D[00000000] mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 00000153 B8[13000000] mov eax, msg2
46 00000158 E8B2FFFFFF call sprint
47 0000015D A1[00000000] mov eax,[max]
48 00000162 E81FFFFFFF call iprintLF
49 00000167 E86FFFFFFF call quit
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Переместить 7Поиск 8Удалить 9МенюМС 10Выход
```

Рис. 2.11: Файл листинга с ошибкой lab7-2

Объектный файл не смог создаться из-за ошибки. Но получился листинг, где выделено место ошибки.

## 2.3 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 № 6. Создайте исполняемый файл и проверьте его работу (рис. [2.12]) (рис. [2.13])

Мой вариант 11 - числа: 21,28,34

```
mc [ramanyyew@fedora]:~/work/arch...
prog-1.asm [----] 0 L: [ 36+31 67/ 68] *(935[*] [X]

    mov eax,msgC
    call sprint
    mov ecx,C
    mov edx,80
    call sread.
    mov eax,C
    call atoi
    mov [C],eax...
....
    mov ecx,[A]
    mov [min],ecx

    cmp ecx, [B]
    jl check_C
    mov ecx, [B]
    mov [min], ecx

check_C:
    cmp ecx, [C]
    jl finish
    mov ecx,[C]
    mov [min],ecx.

finish:
    mov eax,answer
    call sprint

    mov eax, [min]
    call iprintLF

    call quit

1Поч~шь 2Со~ть 3Блок 4За~на 5Копия 6Пе~ть 7Поиск 8Уд~ть
```

Рис. 2.12: Код программы prog-1.asm

```

[ramanyyew@fedora lab07]$
[ramanyyew@fedora lab07]$ nasm -f elf prog-1.asm
[ramanyyew@fedora lab07]$ ld -m elf_i386 prog-1.o -o prog-1
[ramanyyew@fedora lab07]$ ./prog-1
Input A: 21
Input B: 28
Input C: 34
Smallest: 21
[ramanyyew@fedora lab07]$
[ramanyyew@fedora lab07]$

```

Рис. 2.13: Компиляция и запуск программы prog-1.asm

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. (рис. [2.14]) (рис. [2.15])

Мой вариант 11

$$\begin{cases} 4a, x = 0 \\ 4a + x, x \neq 0 \end{cases}$$

Если подставить  $x = 0, a = 3$ , тогда  $f(x) = 12$

Если подставить  $x = 1, a = 2$ , тогда  $f(x) = 9$

```
mc [ramanyew@fedora]:~/work/arch...
prog-2.asm [----] 11 L:[ 21+27 48/ 53] *(665[*])[X]
    call atoi.
    mov [A],eax

    mov eax,msgX
    call sprintf
    mov ecx,X
    mov edx,80
    call sread.
    mov eax,X
    call atoi
    mov [X],eax...

    mov edx,[X]
    mov ebx,0
    cmp ebx,edx
    je first
    jmp second

first:
    mov eax,[A]
    mov ebx,4
    mul ebx
    call iprintLF.
    call quit

second:
    mov eax,[A]
    mov ebx,4
    mul ebx
    mov ebx,[X]
    add eax,ebx
    call iprintLF.
    call quit
```

1По~щъ 2Со~ть 3Блок 4За~на 5Копия 6Пе~ть 7Поиск 8Уд~ть

Рис. 2.14: Код программы prog-2.asm

```
[ramanyyew@fedora lab07]$  
[ramanyyew@fedora lab07]$ nasm -f elf prog-2.asm  
[ramanyyew@fedora lab07]$ ld -m elf_i386 prog-2.o -o prog-2  
[ramanyyew@fedora lab07]$ ./prog-2  
Input A: 3  
Input X: 0  
12  
[ramanyyew@fedora lab07]$ ./prog-2  
Input A: 2  
Input X: 1  
9  
[ramanyyew@fedora lab07]$
```

Рис. 2.15: Компиляция и запуск программы prog-2.asm

## 3 Выводы

Изучили команды условного и безусловного переходов, познакомились с фалом листинга.