

# **Отчёт по лабораторной работе 7**

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

Гадаборшев Заур Закреевич НПИбд-01-23

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Теоретическое введение</b>	<b>6</b>
2.1	Команды перехода . . . . .	6
2.2	Листинг . . . . .	7
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
3.1	Реализация переходов в NASM . . . . .	8
3.2	Изучение структуры файлы листинга . . . . .	16
3.3	Задание для самостоятельной работы . . . . .	19
<b>4</b>	<b>Выводы</b>	<b>24</b>

## Список иллюстраций

3.1	Программа в файле lab7-1.asm . . . . .	9
3.2	Запуск программы lab7-1.asm . . . . .	10
3.3	Программа в файле lab7-1.asm . . . . .	11
3.4	Запуск программы lab7-1.asm . . . . .	12
3.5	Программа в файле lab7-1.asm . . . . .	13
3.6	Запуск программы lab7-1.asm . . . . .	14
3.7	Программа в файле lab7-2.asm . . . . .	15
3.8	Запуск программы lab7-2.asm . . . . .	16
3.9	Файл листинга lab7-2 . . . . .	17
3.10	Ошибка трансляции lab7-2 . . . . .	18
3.11	Файл листинга с ошибкой lab7-2 . . . . .	19
3.12	Программа в файле task7-1.asm . . . . .	20
3.13	Запуск программы task7-1.asm . . . . .	21
3.14	Программа в файле task2.asm . . . . .	22
3.15	Запуск программы task2.asm . . . . .	23

## 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 Теоретическое введение

### 2.1 Команды перехода

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.

Безусловный переход выполняется инструкцией `jmp` (от англ. `jump` – прыжок), которая включает в себя адрес перехода, куда следует передать управление.

Как отмечалось выше, для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

Инструкция `cmp` является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция `cmp` является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания.

## 2.2 Листинг

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

Итак, структура 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);
- адрес — это смещение машинного кода от начала текущего сегмента;
- машинный код представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности. (например, инструкция `int 80h` начинается по смещению `00000020` в сегменте кода; далее идёт машинный код, в который ассемблируется инструкция, то есть инструкция `int 80h` ассемблируется в `CD80` (в шестнадцатеричном представлении); `CD80` — это инструкция на машинном языке, вызывающая прерывание ядра)
- исходный текст программы — это просто строка исходной программы вместе с комментариями (некоторые строки на языке ассемблера, например, строки, содержащие только комментарии, не генерируют никакого машинного кода, и поля «смещение» и «исходный текст программы» в таких строках отсутствуют, однако номер строки им присваивается)

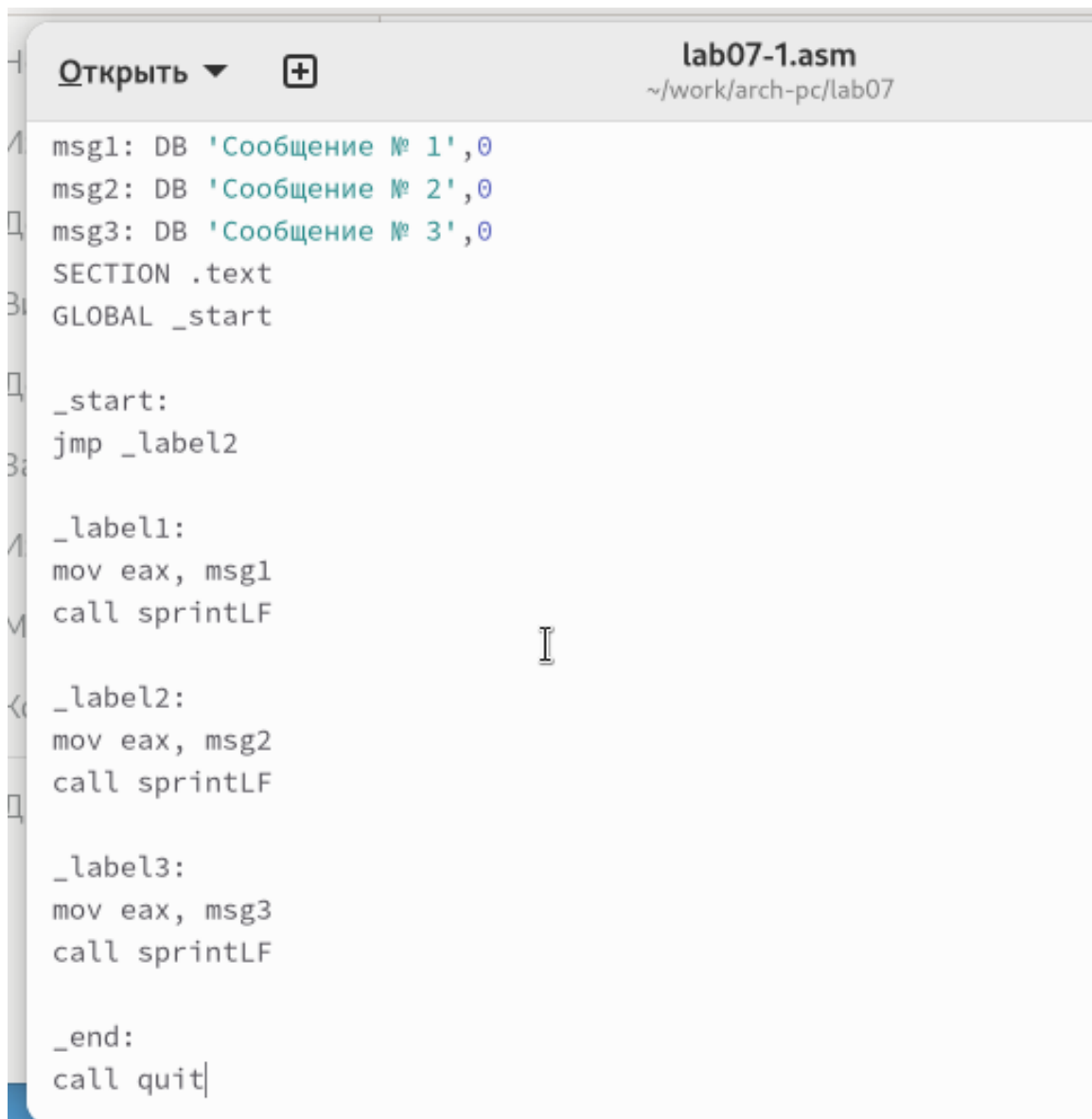
## 3 Выполнение лабораторной работы

### 3.1 Реализация переходов в NASM

Создал каталог для программам лабораторной работы № 7 и файл lab7-1.asm

Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Написал в файл lab7-1.asm текст программы из листинга 7.1.





```
lab07-1.asm
~/work/arch-pc/lab07

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

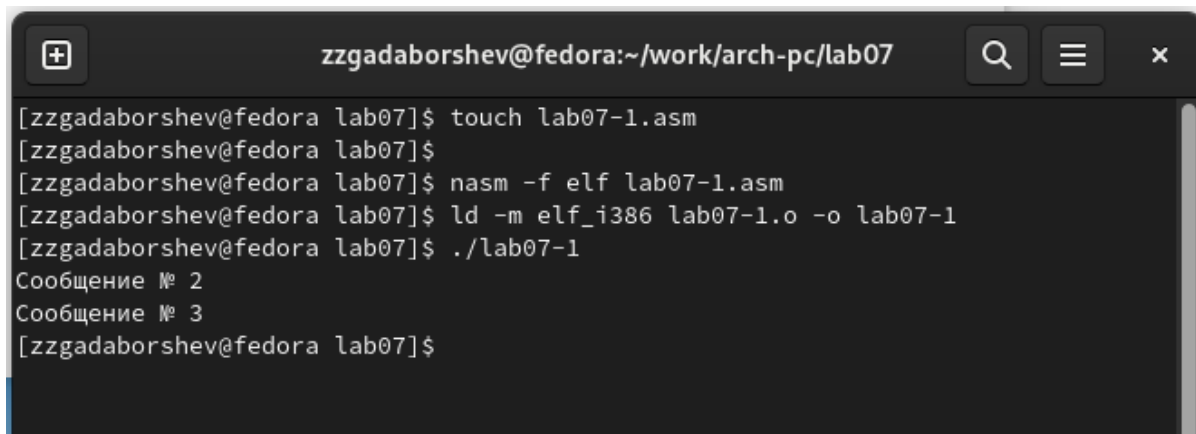
_label2:
mov eax, msg2
call sprintf

_label3:
mov eax, msg3
call sprintf

_end:
call quit
```

Рис. 3.1: Программа в файле lab7-1.asm

Создал исполняемый файл и запустил его.

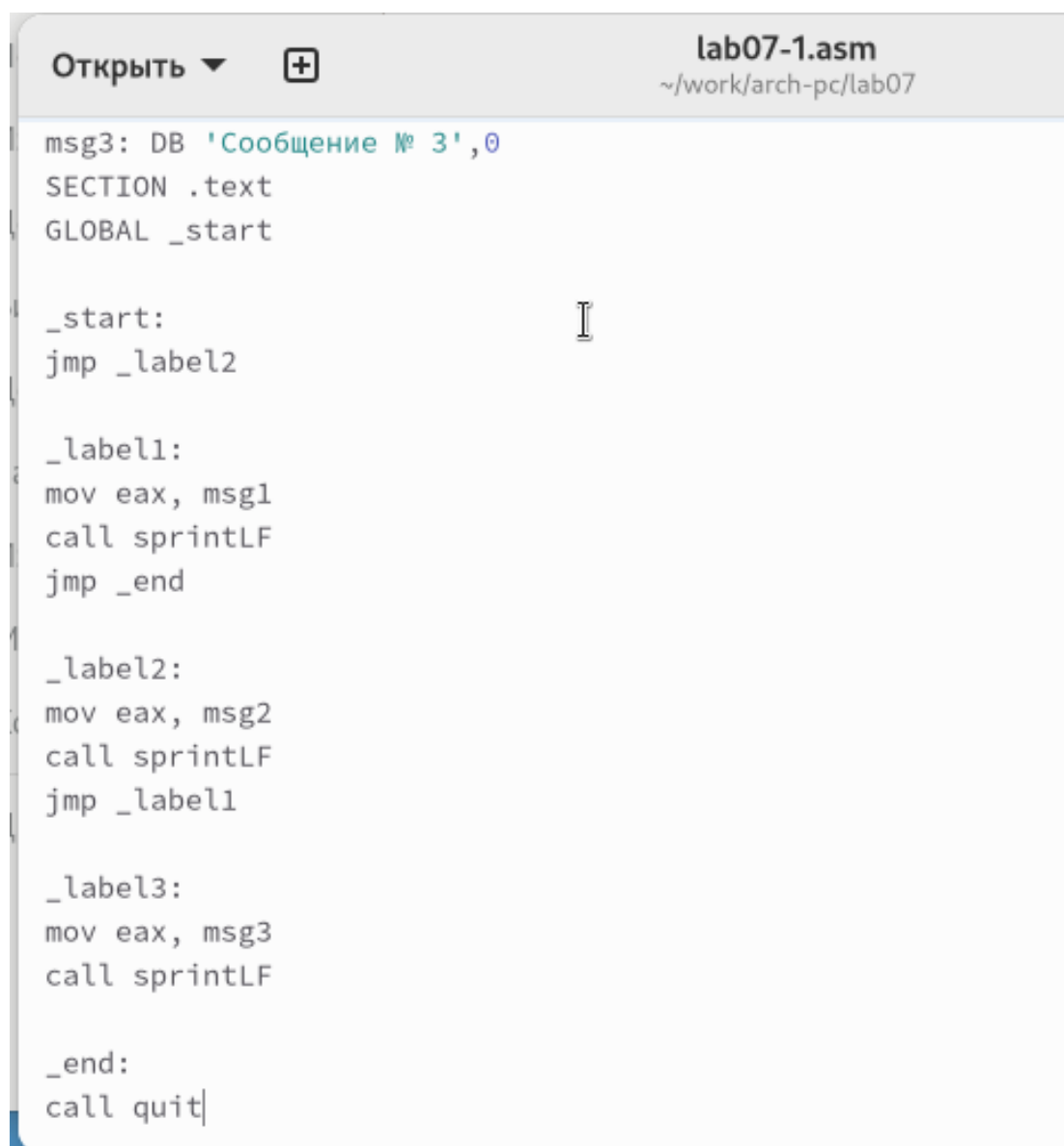
A terminal window with a dark background. The title bar shows the user 'zzgadaborshev@fedora' and the directory '~/work/arch-pc/lab07'. The terminal contains the following text:

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

Рис. 3.2: Запуск программы lab7-1.asm

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`).

Изменил текст программы в соответствии с листингом 7.2.



```
lab07-1.asm
~/work/arch-pc/lab07

Открыть ▼ +

msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
    jmp _label2

_label1:
    mov eax, msg1
    call sprintfLF
    jmp _end

_label2:
    mov eax, msg2
    call sprintfLF
    jmp _label1

_label3:
    mov eax, msg3
    call sprintfLF

_end:
    call quit
```

Рис. 3.3: Программа в файле lab7-1.asm

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

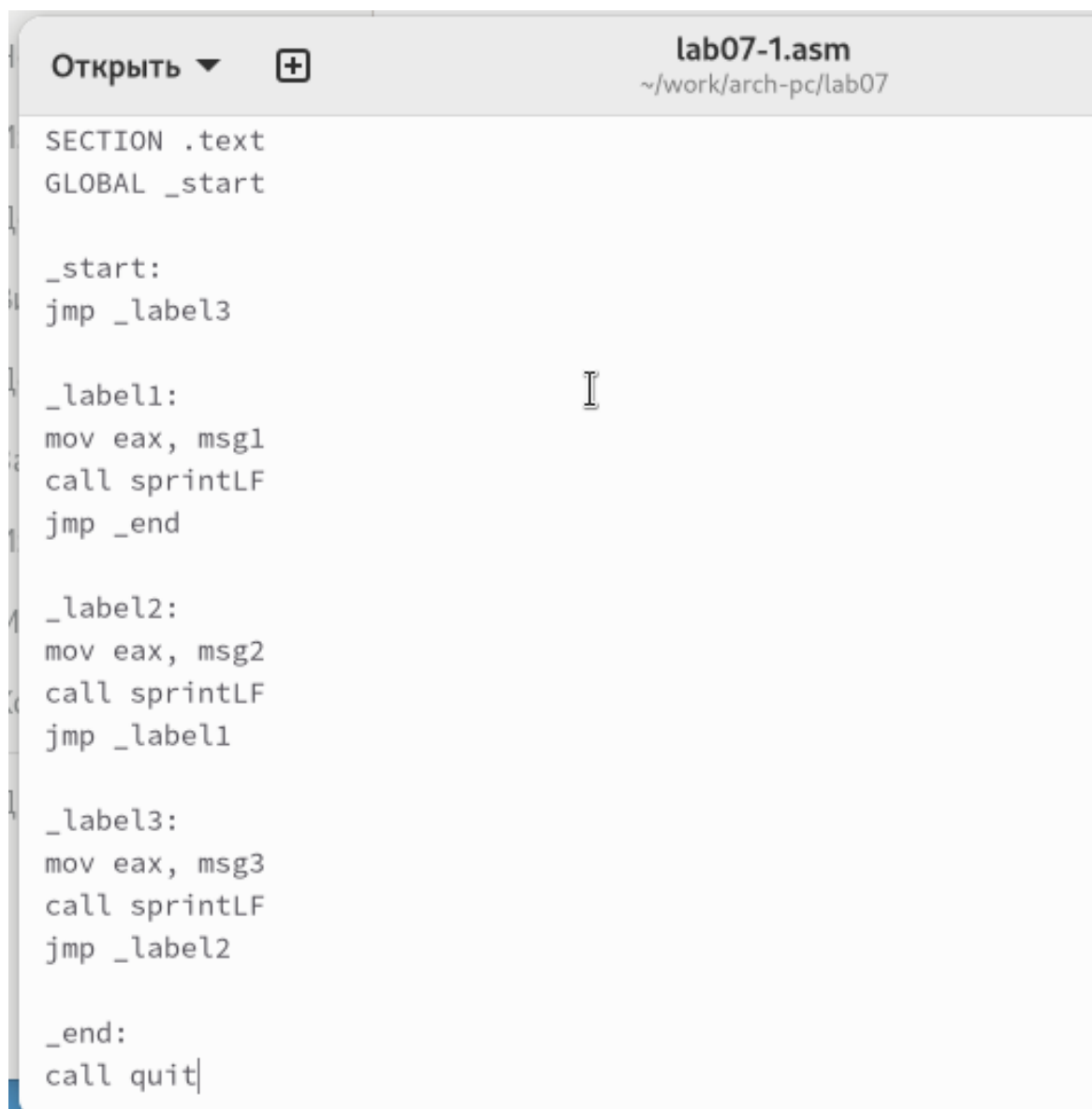
Рис. 3.4: Запуск программы lab7-1.asm

Изменил текст программы, изменив инструкции `jmp`, чтобы вывод программы был следующим:

Сообщение № 3

Сообщение № 2

Сообщение № 1





```
Открыть ▾  lab07-1.asm  
~/work/arch-pc/lab07  
  
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
```

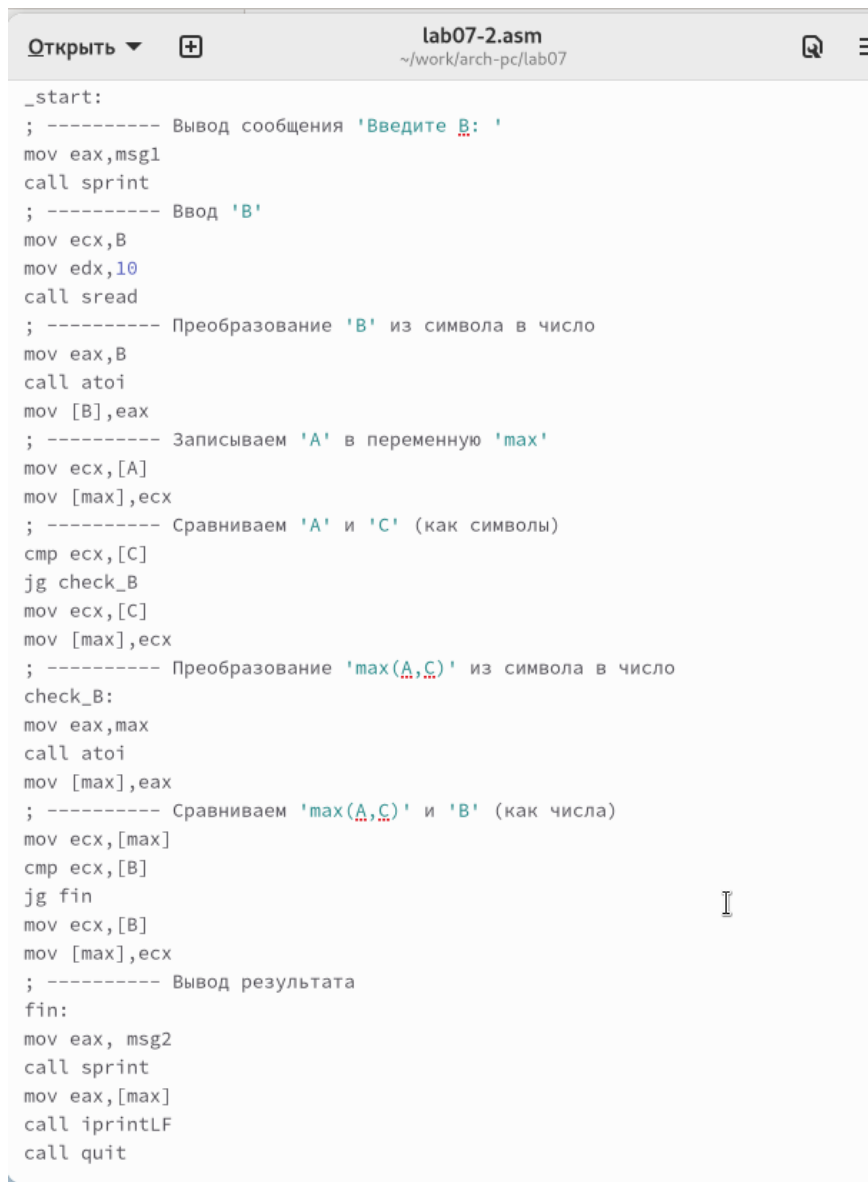
Рис. 3.5: Программа в файле lab7-1.asm

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

Рис. 3.6: Запуск программы lab7-1.asm

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.

Создал исполняемый файл и проверил его работу для разных значений В.



```
Открыть + lab07-2.asm ~/work/arch-pc/lab07
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
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
```

Рис. 3.7: Программа в файле lab7-2.asm

```
[zzgadaborshev@fedora lab07]$  
[zzgadaborshev@fedora lab07]$ touch lab07-2.asm  
[zzgadaborshev@fedora lab07]$ nasm -f elf lab07-2.asm  
[zzgadaborshev@fedora lab07]$ ld -m elf_i386 lab07-2.o -o lab07-2  
[zzgadaborshev@fedora lab07]$ ./lab07-2  
Введите B: 20  
Наибольшее число: 50  
[zzgadaborshev@fedora lab07]$ ./lab07-2  
Введите B: 30  
Наибольшее число: 50  
[zzgadaborshev@fedora lab07]$ ./lab07-2  
Введите B: 60  
Наибольшее число: 60  
[zzgadaborshev@fedora lab07]$
```

Рис. 3.8: Запуск программы lab7-2.asm

## 3.2 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`



```

Открыть  + lab07-2.lst Стр. 189, Поз. 5
~/work/arch-pc/lab07

187 12      _start:
188 13      ; ----- Вывод сообщения 'Введите B: '
189 14 000000E8 B8[00000000]    mov eax,msg1
190 15 000000ED E81DFFFFFF    call sprint
191 16      ; ----- Ввод 'B'
192 17 000000F2 B0[0A000000]    mov ecx,B
193 18 000000F7 BA0A000000    mov edx,10
194 19 000000FC E842FFFFFF    call read
195 20      ; ----- Преобразование 'B' из символа в число
196 21 00000101 B8[0A000000]    mov eax,B
197 22 00000106 E891FFFFFF    call atoi
198 23 0000010B A3[0A000000]    mov [B],eax
199 24      ; ----- Записываем 'A' в переменную 'max'
200 25 00000110 8B0D[35000000]    mov ecx,[A]
201 26 00000116 890D[00000000]    mov [max],ecx
202 27      ; ----- Сравниваем 'A' и 'C' (как символы)
203 28 0000011C 3B0D[39000000]    cmp ecx,[C]
204 29 00000122 7F0C          jg check_B
205 30 00000124 8B0D[39000000]    mov ecx,[C]
206 31 0000012A 890D[00000000]    mov [max],ecx
207 32      ; ----- Преобразование 'max(A,C)' из символа в число
208 33      check_B:
209 34 00000130 B8[00000000]    mov eax,max
210 35 00000135 E862FFFFFF    call atoi
211 36 0000013A A3[00000000]    mov [max],eax
212 37      ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
213 38 0000013F 8B0D[00000000]    mov ecx,[max]
214 39 00000145 3B0D[0A000000]    cmp ecx,[B]
215 40 0000014B 7F0C          jg fin
216 41 0000014D 8B0D[0A000000]    mov ecx,[B]
217 42 00000153 890D[00000000]    mov [max],ecx
218 43      ; ----- Вывод результата
219 44      fin:
220 45 00000159 B8[13000000]    mov eax,msg2
221 46 0000015E E8ACFFFFFF    call sprint
222 47 00000163 A1[00000000]    mov eax,[max]
223 48 00000168 E819FFFFFF    call iprintLF
224 49 0000016D E869FFFFFF    call quit

```

Рис. 3.9: Файл листинга lab7-2

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

строка 203

- 28 - номер строки в подпрограмме
- 0000011C - адрес
- 3B0D[39000000] - машинный код
- `cmp ecx,[C]` - код программы - сравнивает регистр `ecx` и переменную `C`

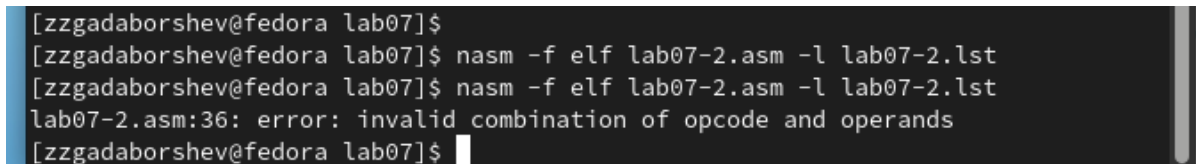
строка 204

- 29 - номер строки в подпрограмме
- 00000122 - адрес
- 7F0C - машинный код
- `jb check_B` - код программы - если `>`, то переход к метке `check_B`

строка 205

- 30 - номер строки в подпрограмме
- 00000124 - адрес
- `8B0D[39000000]` - машинный код
- `mov esx,[C]` - код программы - перекладывает в регистр `esx` значение переменной `C`

Открыл файл с программой `lab7-2.asm` и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга.



```
[zzgadaborshev@fedora lab07]$
[zzgadaborshev@fedora lab07]$ nasm -f elf lab07-2.asm -l lab07-2.lst
[zzgadaborshev@fedora lab07]$ nasm -f elf lab07-2.asm -l lab07-2.lst
lab07-2.asm:36: error: invalid combination of opcode and operands
[zzgadaborshev@fedora lab07]$
```

Рис. 3.10: Ошибка трансляции `lab7-2`

```
lab07-2.asm lab07-2.lst
190 15 000000ED E81DFFFFFF call sprint
191 16 ; ----- Ввод 'B'
192 17 000000F2 B9[0A000000] mov ecx,B
193 18 000000F7 BA0A000000 mov edx,10
194 19 000000FC E842FFFFFF call sread
195 20 ; ----- Преобразование 'B' из символа в число
196 21 00000101 B8[0A000000] mov eax,B
197 22 00000106 E801FFFFFF call atoi
198 23 0000010B A3[0A000000] mov [B],eax
199 24 ; ----- Записываем 'A' в переменную 'max'
200 25 00000110 8B0D[35000000] mov ecx,[A]
201 26 00000116 890D[00000000] mov [max],ecx
202 27 ; ----- Сравниваем 'A' и 'C' (как символы)
203 28 0000011C 3B0D[39000000] cmp ecx,[C]
204 29 00000122 7F0C jg check_B
205 30 00000124 8B0D[39000000] mov ecx,[C]
206 31 0000012A 890D[00000000] mov [max],ecx
207 32 ; ----- Преобразование 'max(A,C)' из символа в число
208 33 check_B:
209 34 00000130 B8[00000000] mov eax,max
210 35 00000135 E862FFFFFF call atoi
211 36 mov [max],
212 36 ***** error: invalid combination of opcode and operands
213 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
214 38 0000013A 8B0D[00000000] mov ecx,[max]
215 39 00000140 3B0D[0A000000] cmp ecx,[B]
216 40 00000146 7F0C jg fin
217 41 00000148 8B0D[0A000000] mov ecx,[B]
218 42 0000014E 890D[00000000] mov [max],ecx
219 43 ; ----- Вывод результата
220 44 fin:
221 45 00000154 B8[13000000] mov eax,msg2
222 46 00000159 E8B1FFFFFF call sprint
223 47 0000015E A1[00000000] mov eax,[max]
224 48 00000163 E81EFFFFFF call iprintLF
225 49 00000168 E86EFFFFFF call quit
```

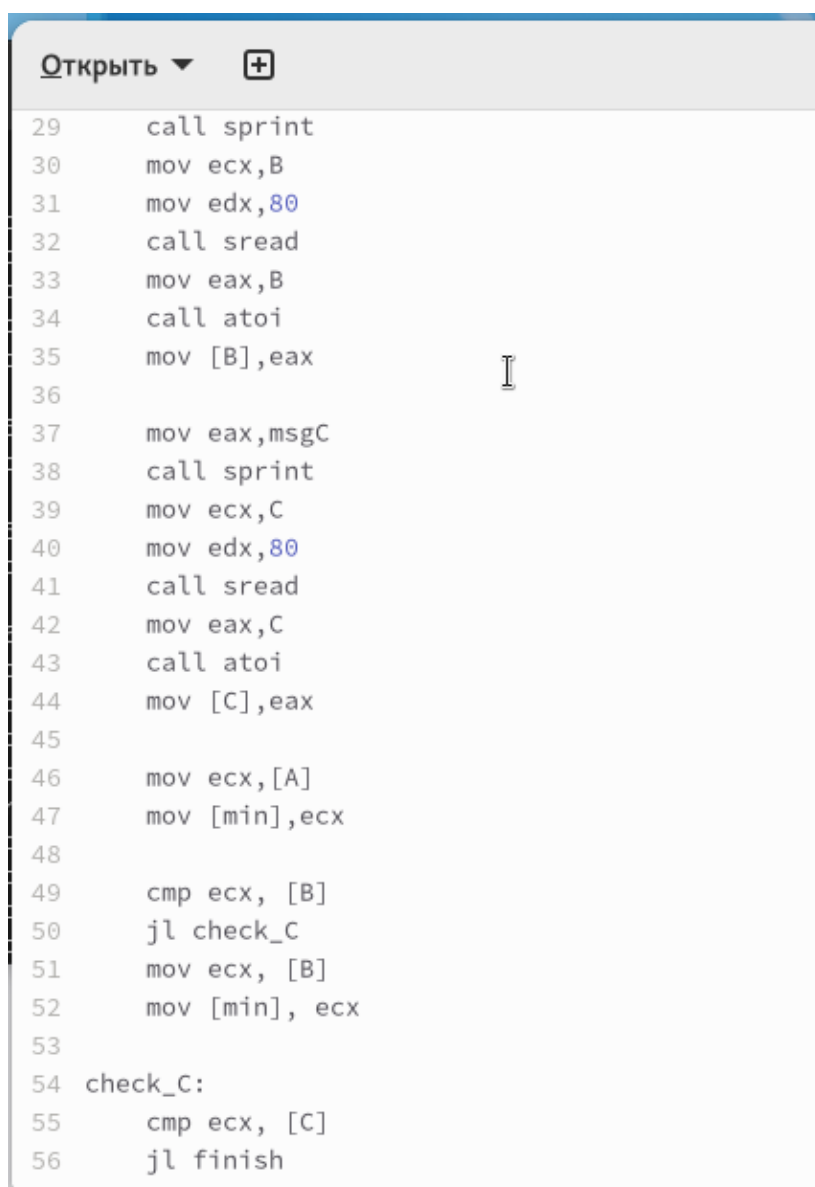
Рис. 3.11: Файл листинга с ошибкой lab7-2

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

### 3.3 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

для варианта 14 - 81, 22, 72



```
29    call sprint
30    mov ecx,B
31    mov edx,80
32    call sread
33    mov eax,B
34    call atoi
35    mov [B],eax
36
37    mov eax,msgC
38    call sprint
39    mov ecx,C
40    mov edx,80
41    call sread
42    mov eax,C
43    call atoi
44    mov [C],eax
45
46    mov ecx,[A]
47    mov [min],ecx
48
49    cmp ecx, [B]
50    jl check_C
51    mov ecx, [B]
52    mov [min], ecx
53
54 check_C:
55    cmp ecx, [C]
56    jl finish
57    mov ecx, [C]
```

Рис. 3.12: Программа в файле task7-1.asm

```
[zzgadaborshev@fedora lab07]$  
[zzgadaborshev@fedora lab07]$ touch task7-1.asm  
[zzgadaborshev@fedora lab07]$ nasm -f elf task7-1.asm  
[zzgadaborshev@fedora lab07]$ ld -m elf_i386 task7-1.o -o task7-1  
[zzgadaborshev@fedora lab07]$ ./task7-1  
Input A: 81  
Input B: 22  
Input C: 72  
Smallest: 22  
[zzgadaborshev@fedora lab07]$  
[zzgadaborshev@fedora lab07]$
```

Рис. 3.13: Запуск программы task7-1.asm

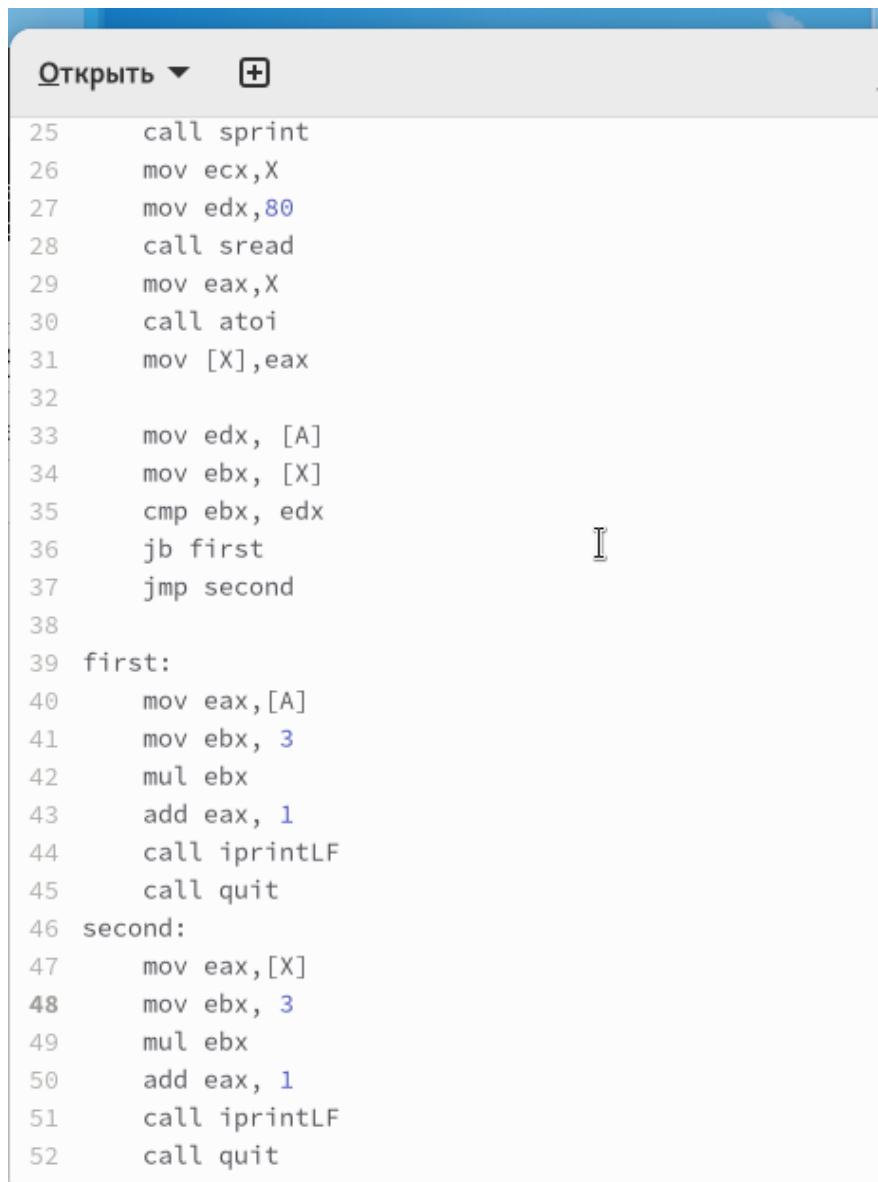
Напишите программу, которая для введенных с клавиатуры значений  $x$  и  $a$  вычисляет значение заданной функции  $f(x)$  и выводит результат вычислений. Вид функции  $f(x)$  выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений  $X$  и  $a$  из 7.6.

для варианта 14

$$\begin{cases} 3a + 1, x < a \\ 3x + 1, x \geq a \end{cases}$$

Если подставить  $x = 2, a = 3$  получается  $3 * 3 + 1 = 10$ .

Если подставить  $x = 4, a = 2$  получается  $3 * 4 + 1 = 13$ .



```
Открыть ▾ +
25    call sprint
26    mov ecx,X
27    mov edx,80
28    call sread
29    mov eax,X
30    call atoi
31    mov [X],eax
32
33    mov edx, [A]
34    mov ebx, [X]
35    cmp ebx, edx
36    jb first
37    jmp second
38
39 first:
40    mov eax,[A]
41    mov ebx, 3
42    mul ebx
43    add eax, 1
44    call iprintLF
45    call quit
46 second:
47    mov eax,[X]
48    mov ebx, 3
49    mul ebx
50    add eax, 1
51    call iprintLF
52    call quit
```

Рис. 3.14: Программа в файле task2.asm

```
[zzgadaborshev@fedora lab07]$ touch task7-2.asm
[zzgadaborshev@fedora lab07]$
[zzgadaborshev@fedora lab07]$ nasm -f elf task7-2.asm
[zzgadaborshev@fedora lab07]$ ld -m elf_i386 task7-2.o -o task7-2
[zzgadaborshev@fedora lab07]$ ./task7-2
Input A: 3
Input X: 2
10
[zzgadaborshev@fedora lab07]$ ./task7-2
Input A: 2
Input X: 4
13
[zzgadaborshev@fedora lab07]$
```

Рис. 3.15: Запуск программы task2.asm

## 4 Выводы

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