

Отчёт по лабораторной работе 7

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

Татьяна Соколова НММбд-03-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файлы листинга	13
2.3	Самостоятельное задание	16
3	Выводы	21

Список иллюстраций

2.1	Создан каталог	6
2.2	Программа lab7-1.asm	7
2.3	Запуск программы lab7-1.asm	7
2.4	Программа lab7-1.asm	8
2.5	Запуск программы lab7-1.asm	9
2.6	Программа lab7-1.asm	10
2.7	Запуск программы lab7-1.asm	10
2.8	Программа lab7-2.asm	12
2.9	Запуск программы lab7-2.asm	13
2.10	Файл листинга lab7-2	14
2.11	Ошибка трансляции lab7-2	15
2.12	Файл листинга с ошибкой lab7-2	16
2.13	Программа lab7-task1.asm	17
2.14	Запуск программы lab7-task1.asm	17
2.15	Программа lab7-task2.asm	19
2.16	Запуск программы lab7-task2.asm	20

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

Создаю каталог для программ лабораторной работы № 7 и файл lab7-1.asm.
(рис. 2.1)

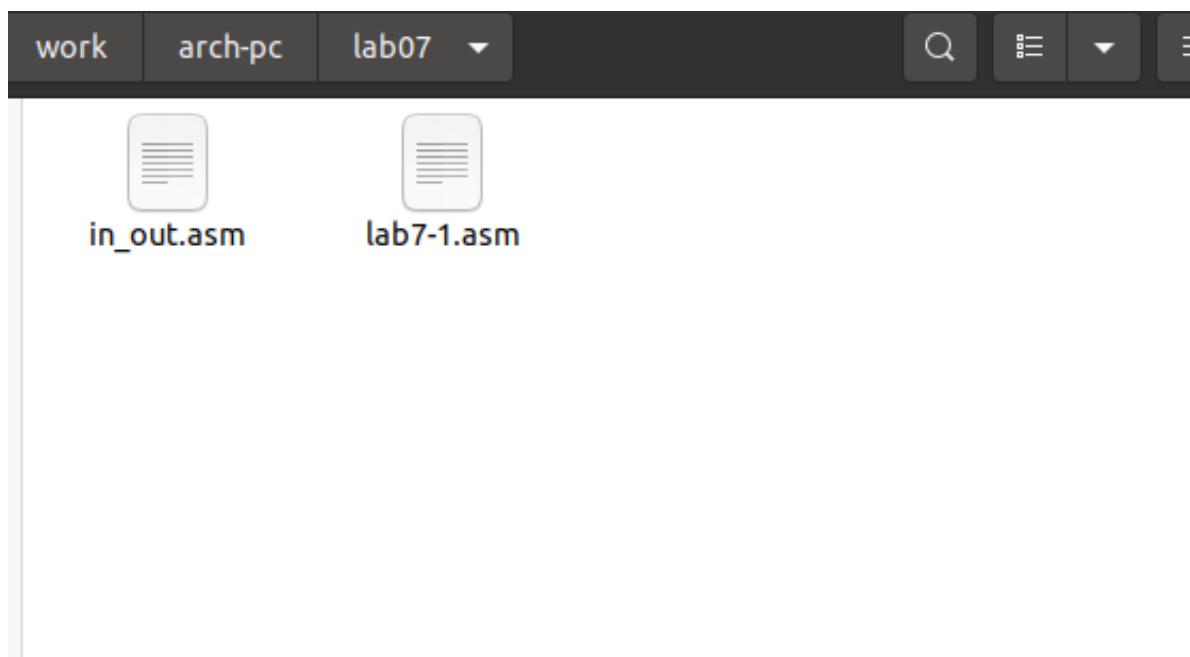
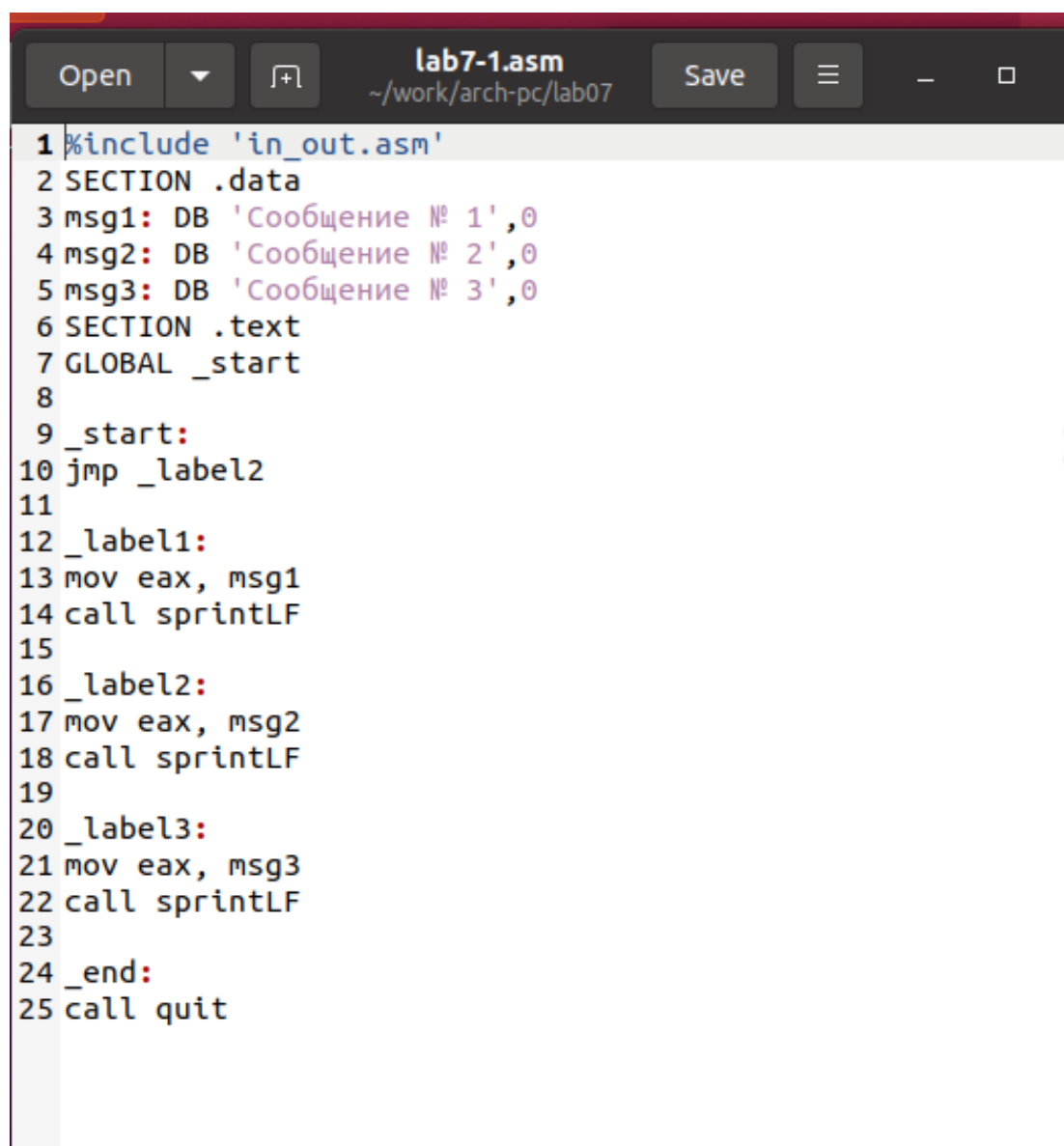


Рис. 2.1: Создан каталог

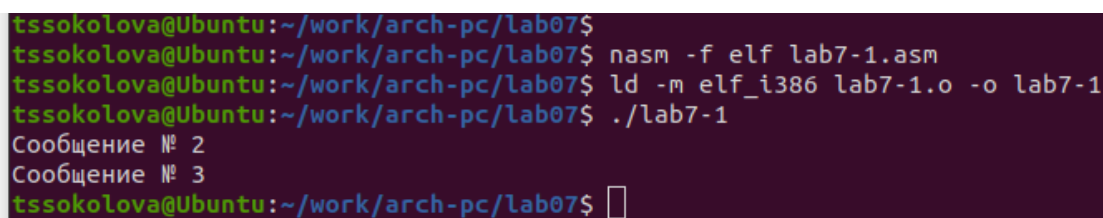
Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Написал в файл `lab7-1.asm` текст программы из листинга 7.1. (рис. 2.2)



```
1 %include 'in_out.asm'
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
9 _start:
10 jmp _label2
11
12 _label1:
13 mov eax, msg1
14 call sprintLF
15
16 _label2:
17 mov eax, msg2
18 call sprintLF
19
20 _label3:
21 mov eax, msg3
22 call sprintLF
23
24 _end:
25 call quit
```

Рис. 2.2: Программа lab7-1.asm

Создаю исполняемый файл и запускаю его. (рис. 2.3)

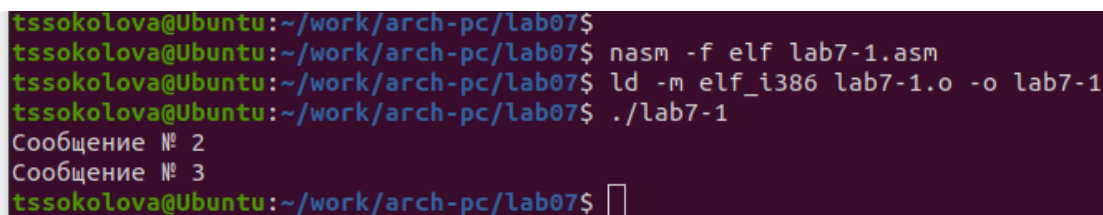


```
tssokolova@Ubuntu:~/work/arch-pc/lab07$
tssokolova@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
tssokolova@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
tssokolova@Ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
tssokolova@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.3: Запуск программы lab7-1.asm

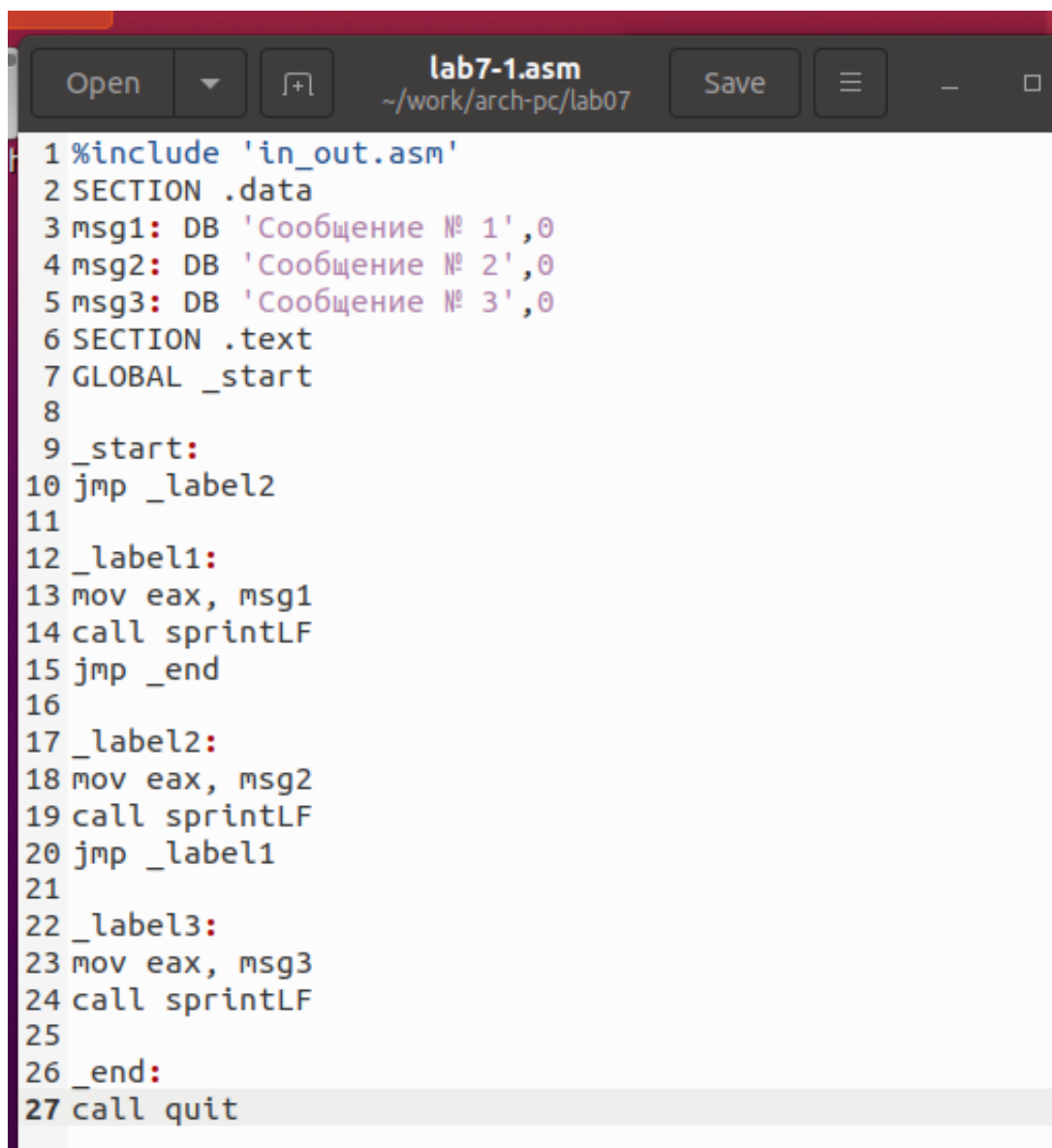
Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`).

Изменяю текст программы в соответствии с листингом 7.2. (рис. 2.4) (рис. 2.5)



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

Рис. 2.4: Программа lab7-1.asm

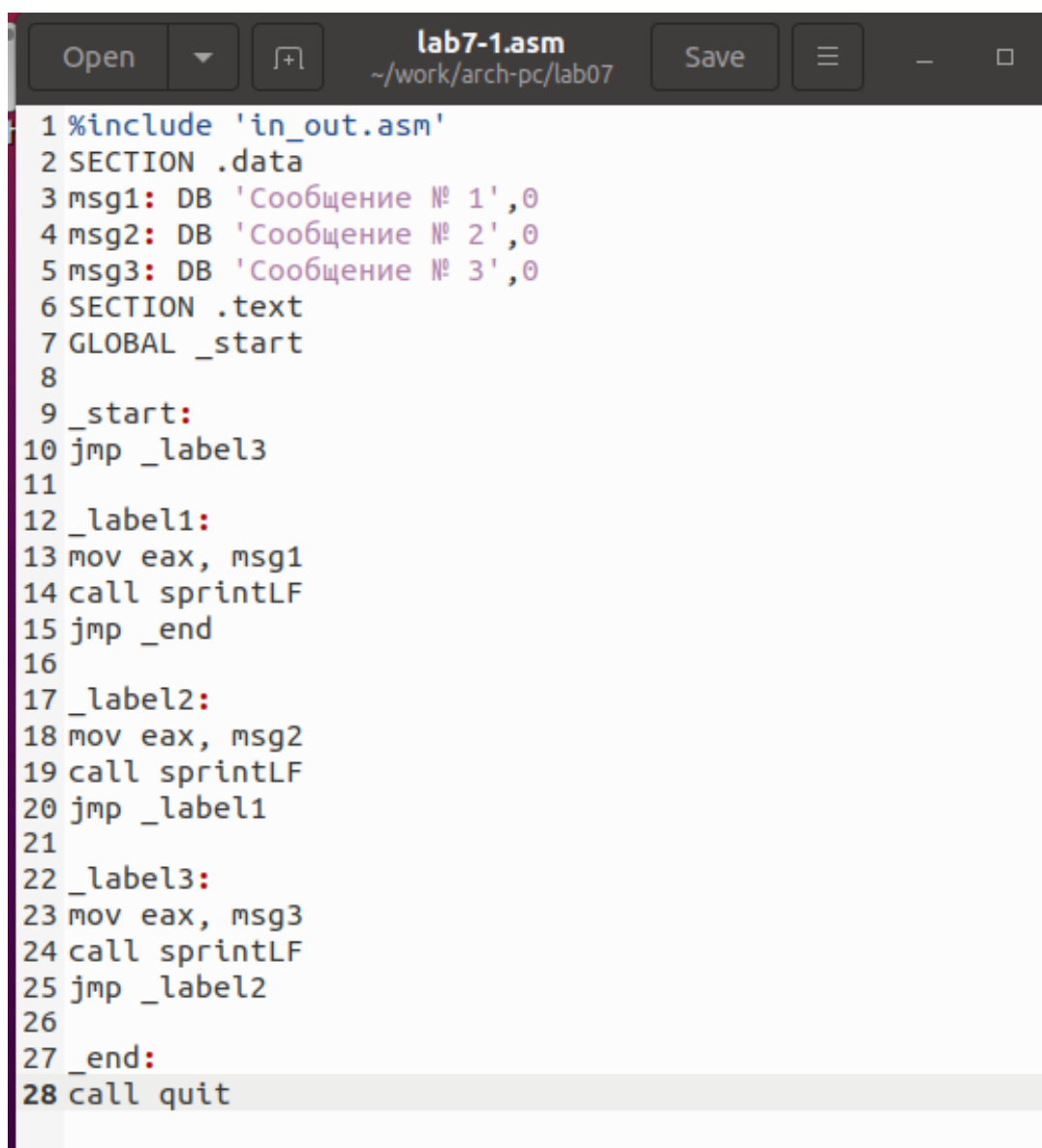


```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8
9 _start:
10 jmp _label2
11
12 _label1:
13 mov eax, msg1
14 call sprintLF
15 jmp _end
16
17 _label2:
18 mov eax, msg2
19 call sprintLF
20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

Рис. 2.5: Запуск программы lab7-1.asm

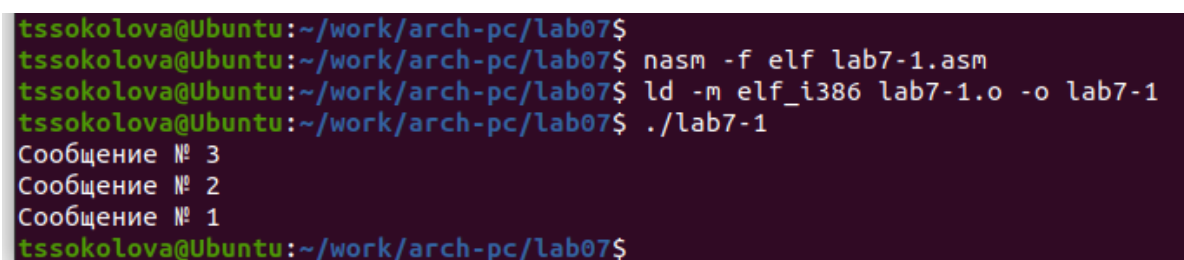
Изменила текст программы, чтобы вывод программы был следующим (рис. 2.6) (рис. 2.7):

Сообщение № 3
Сообщение № 2
Сообщение № 1



```
1 %include 'in_out.asm'
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
9 _start:
10 jmp _label3
11
12 _label1:
13 mov eax, msg1
14 call sprintLF
15 jmp _end
16
17 _label2:
18 mov eax, msg2
19 call sprintLF
20 jmp _label1
21
22 _label3:
23 mov eax, msg3
24 call sprintLF
25 jmp _label2
26
27 _end:
28 call quit
```

Рис. 2.6: Программа lab7-1.asm

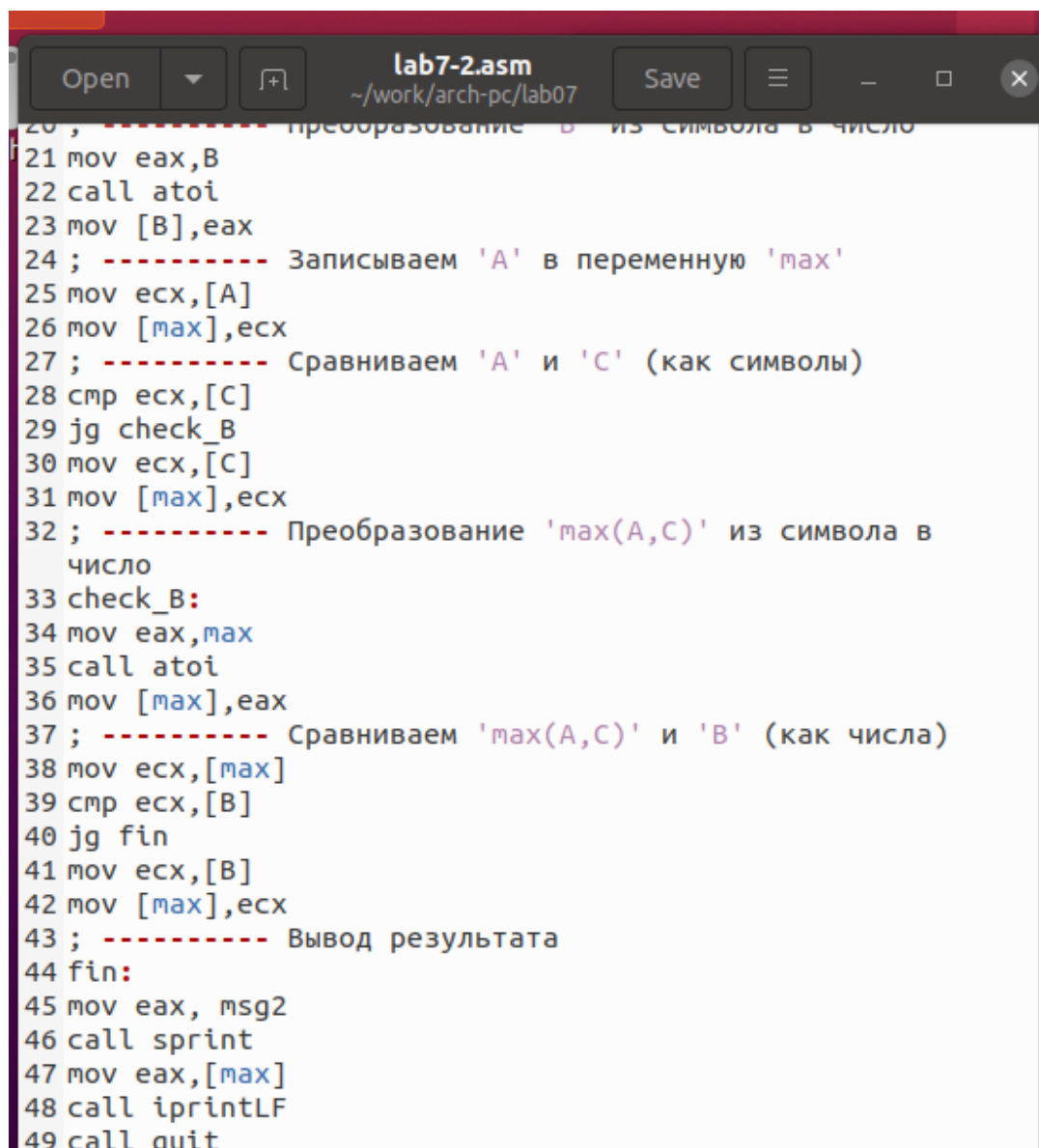


```
tssokolova@Ubuntu:~/work/arch-pc/lab07$
tssokolova@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
tssokolova@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
tssokolova@Ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
tssokolova@Ubuntu:~/work/arch-pc/lab07$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

Рис. 2.7: 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.

Создала исполняемый файл и проверила его работу для разных значений В (рис. 2.8) (рис. 2.9).



```
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,B
22 call atoi
23 mov [B],eax
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A]
26 mov [max],ecx
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C]
29 jg check_B
30 mov ecx,[C]
31 mov [max],ecx
32 ; ----- Преобразование 'max(A,C)' из символа в
    число
33 check_B:
34 mov eax,max
35 call atoi
36 mov [max],eax
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 mov ecx,[max]
39 cmp ecx,[B]
40 jg fin
41 mov ecx,[B]
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 mov eax, msg2
46 call sprint
47 mov eax,[max]
48 call iprintLF
49 call quit
```

Рис. 2.8: Программа lab7-2.asm

```
tssokolova@Ubuntu:~/work/arch-pc/lab07$  
tssokolova@Ubuntu:~/work/arch-pc/lab07$  
tssokolova@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm  
tssokolova@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2  
tssokolova@Ubuntu:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 6  
Наибольшее число: 50  
tssokolova@Ubuntu:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 56  
Наибольшее число: 56  
tssokolova@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.9: 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.

Создаю файл листинга для программы из файла lab7-2.asm (рис. 2.10)

```

lab7-2.asm
200      25 00000110 8B0D[35000000]
201      26 00000116 890D[00000000]
202      27
      'C' (как символы)
203      28 0000011C 3B0D[39000000]
204      29 00000122 7F0C
205      30 00000124 8B0D[39000000]
206      31 0000012A 890D[00000000]
207      32
      'max(A,C)' из символа в число
208      33
209      34 00000130 B8[00000000]
210      35 00000135 E862FFFFFF
211      36 0000013A A3[00000000]
212      37
      'max(A,C)' и 'B' (как числа)
213      38 0000013F 8B0D[00000000]
214      39 00000145 3B0D[0A000000]
215      40 0000014B 7F0C
216      41 0000014D 8B0D[0A000000]
217      42 00000153 890D[00000000]
218      43
219      44
220      45 00000159 B8[13000000]
221      46 0000015E E8ACFFFFFF
222      47 00000163 A1[00000000]
223      48 00000168 E819FFFFFF
224      49 0000016D E869FFFFFF

lab7-2.lst
mov ecx,[A]
mov [max],ecx
; ----- Сравниваем 'A' и
cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx
; ----- Преобразование
check_B:
mov eax,max
call atoi
mov [max],eax
; ----- Сравниваем
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. 2.10: Файл листинга lab7-2

Ознакомимся с его форматом и содержимым.

строка 211

- 34 - номер строки
- 0000012E - адрес
- B8[00000000] - машинный код
- mov eax,max - код программы

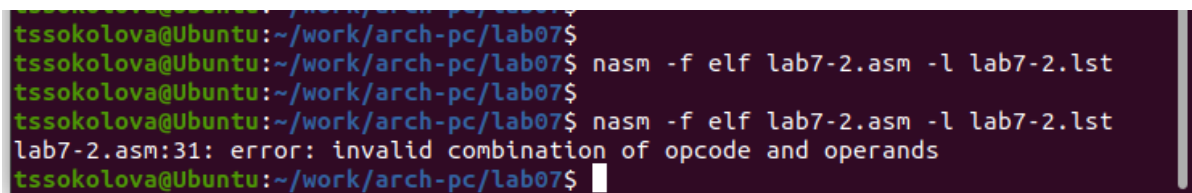
строка 212

- 35 - номер строки
- 00000133 - адрес
- E864FFFFFF - машинный код
- call atoi - код программы

строка 213

- 36 - номер строки
- 00000138 - адрес
- A3[00000000] - машинный код
- mov [max],eax - код программы

Открыла файл с программой lab7-2.asm и в инструкции с двумя операндами удалила один операнд. Выполню трансляцию с получением файла листинга. (рис. 2.11) (рис. 2.12)



```
tssokolova@Ubuntu:~/work/arch-pc/lab07$
tssokolova@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
tssokolova@Ubuntu:~/work/arch-pc/lab07$
tssokolova@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:31: error: invalid combination of opcode and operands
tssokolova@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.11: Ошибка трансляции lab7-2

```

lab7-2.asm
201 20 00000110 0900[00000000]
202 27
    'C' (как символы)
203 28 0000011C 3B0D[39000000]
204 29 00000122 7F06
205 30 00000124 8B0D[39000000]
206 31
207 31 *****
    opcode and operands
208 32
    'max(A,C)' из символа в число
209 33
210 34 0000012A B8[00000000]
211 35 0000012F E868FFFFFF
212 36 00000134 A3[00000000]
213 37
    и 'B' (как числа)
214 38 00000139 8B0D[00000000]
215 39 0000013F 3B0D[0A000000]
216 40 00000145 7F0C
217 41 00000147 8B0D[0A000000]
218 42 0000014D 890D[00000000]
219 43
220 44
221 45 00000153 B8[13000000]
222 46 00000158 E8B2FFFFFF
223 47 0000015D A1[00000000]
224 48 00000162 E81FFFFFFF
225 49 00000167 E86FFFFFFF

lab7-2.lst
201 20 00000110 0900[00000000]    mov [max],ecx
202 27                               ; ----- Сравниваем 'A' и
    'C' (как символы)
203 28 0000011C 3B0D[39000000]    cmp ecx,[C]
204 29 00000122 7F06              jg check_B
205 30 00000124 8B0D[39000000]    mov ecx,[C]
206 31                            mov [max]
207 31 *****                    error: invalid combination of
    opcode and operands
208 32                               ; ----- Преобразование
    'max(A,C)' из символа в число
209 33
210 34 0000012A B8[00000000]    check_B:
211 35 0000012F E868FFFFFF        mov eax,max
212 36 00000134 A3[00000000]    call atoi
213 37                            mov [max],eax
214 38 00000139 8B0D[00000000]    ; ----- Сравниваем 'max(A,C)'
215 39 0000013F 3B0D[0A000000]    mov ecx,[max]
216 40 00000145 7F0C              cmp ecx,[B]
217 41 00000147 8B0D[0A000000]    jg fin
218 42 0000014D 890D[00000000]    mov ecx,[B]
219 43                            mov [max],ecx
220 44                               ; ----- Вывод результата
221 45 00000153 B8[13000000]    fin:
222 46 00000158 E8B2FFFFFF        mov eax, msg2
223 47 0000015D A1[00000000]    call sprint
224 48 00000162 E81FFFFFFF        mov eax,[max]
225 49 00000167 E86FFFFFFF        call iprintLF
                                   call quit

```

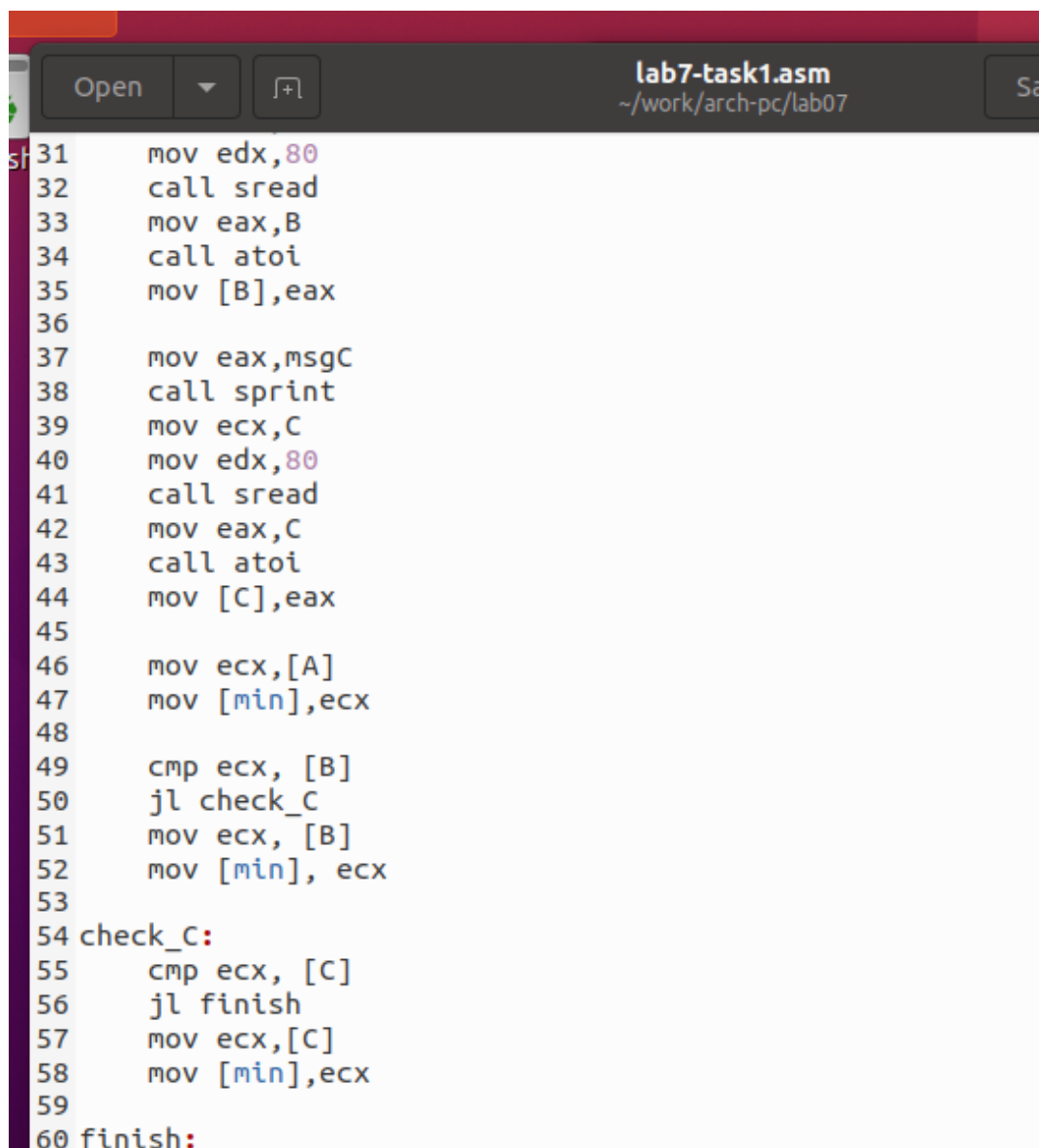
Рис. 2.12: Файл листинга с ошибкой lab7-2

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

2.3 Самостоятельное задание

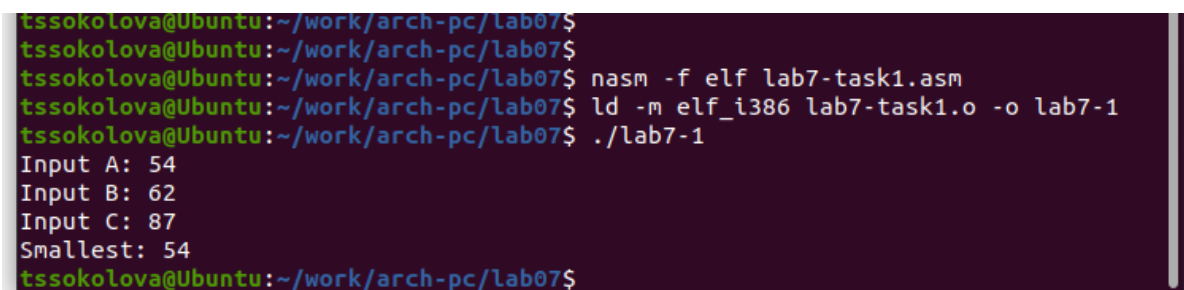
Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу (рис. 2.13) (рис. 2.14)

для варианта 5 - 54,62,87



```
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]
58    mov [min],ecx
59
60 finish:
```

Рис. 2.13: Программа lab7-task1.asm



```
tssokolova@Ubuntu:~/work/arch-pc/lab07$
tssokolova@Ubuntu:~/work/arch-pc/lab07$
tssokolova@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-task1.asm
tssokolova@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-task1.o -o lab7-1
tssokolova@Ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Input A: 54
Input B: 62
Input C: 87
Smallest: 54
tssokolova@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.14: Запуск программы lab7-task1.asm

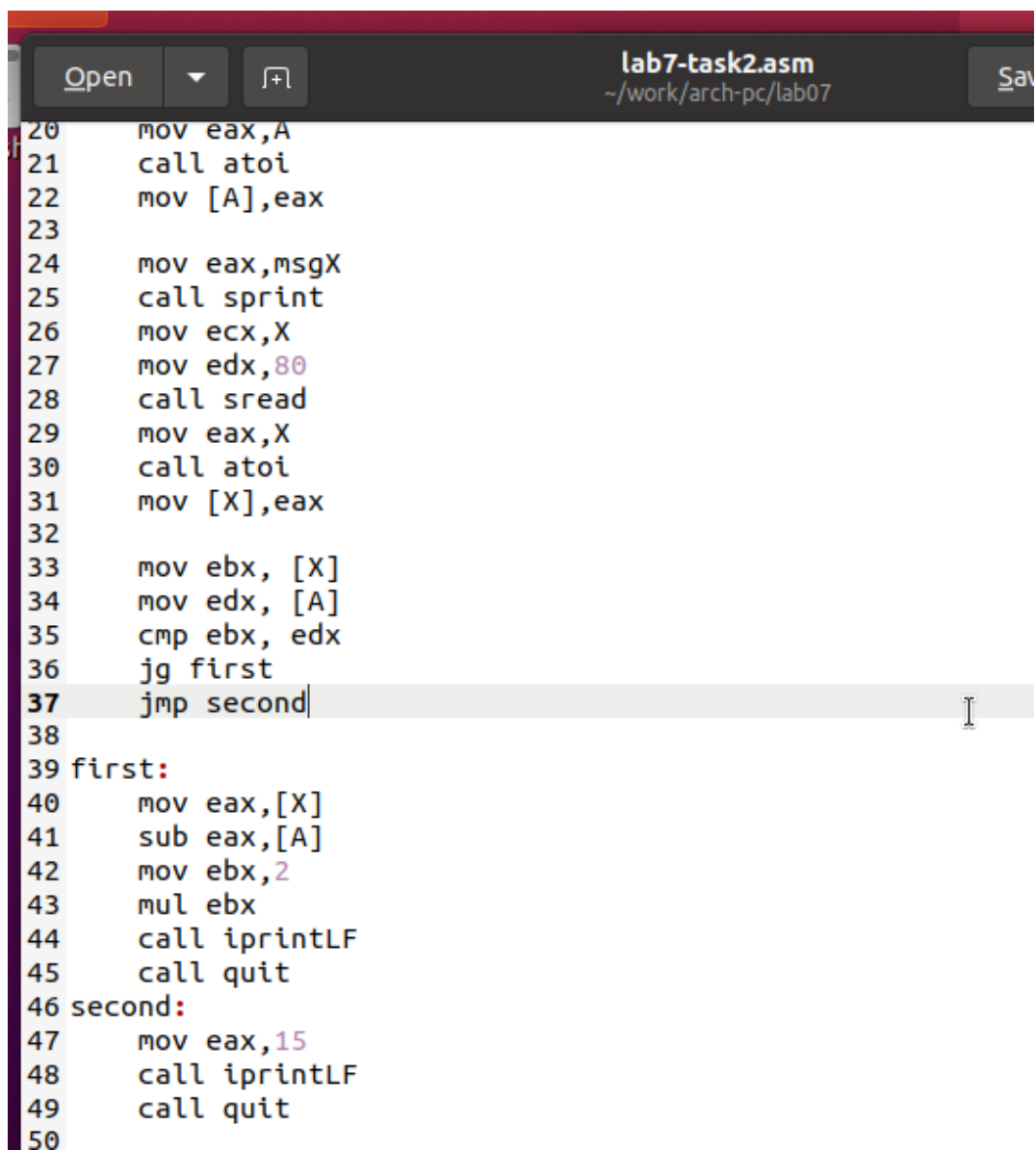
Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений X и a из 7.6. (рис. 2.15) (рис. 2.16)

для варианта 5

$$\begin{cases} 2(x - a), x > a \\ 15, x \leq a \end{cases}$$

При $x = 1, a = 2$ получается 15.

При $x = 2, a = 1$ получается 2.



```
lab7-task2.asm
~/work/arch-pc/lab07

20     mov eax,A
21     call atoi
22     mov [A],eax
23
24     mov eax,msgX
25     call sprintf
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 ebx, [X]
34     mov edx, [A]
35     cmp ebx, edx
36     jg first
37     jmp second
38
39 first:
40     mov eax,[X]
41     sub eax,[A]
42     mov ebx,2
43     mul ebx
44     call iprintLF
45     call quit
46 second:
47     mov eax,15
48     call iprintLF
49     call quit
50
```

Рис. 2.15: Программа lab7-task2.asm

```
tssokolova@Ubuntu:~/work/arch-pc/lab07$  
tssokolova@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-task2.asm  
tssokolova@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-task2.o -o lab7-2  
tssokolova@Ubuntu:~/work/arch-pc/lab07$ ./lab7-2  
Input A: 2  
Input X: 1  
15  
tssokolova@Ubuntu:~/work/arch-pc/lab07$ ./lab7-2  
Input A: 1  
Input X: 2  
2  
tssokolova@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы lab7-task2.asm

3 Выводы

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