

# Отчёт по лабораторной работе 7

Архитектура компьютеров и операционные системы

Ханеков Максат НКА-06-23

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
	2.1 Самостоятельное задание . . . . .	15
3	Выводы	20

## Список иллюстраций

2.1	Создал каталог и файл . . . . .	6
2.2	Программа в файле lab7-1.asm . . . . .	7
2.3	Запуск программы lab7-1.asm . . . . .	8
2.4	Программа в файле lab7-1.asm . . . . .	8
2.5	Запуск программы lab7-1.asm . . . . .	9
2.6	Программа в файле lab7-1.asm . . . . .	10
2.7	Запуск программы lab7-1.asm . . . . .	11
2.8	Программа в файле lab7-2.asm . . . . .	12
2.9	Запуск программы lab7-2.asm . . . . .	13
2.10	Файл листинга lab7-2 . . . . .	13
2.11	Ошибка трансляции lab7-2 . . . . .	15
2.12	Файл листинга с ошибкой lab7-2 . . . . .	15
2.13	Программа в файле lab7-3.asm . . . . .	16
2.14	Запуск программы lab7-3.asm . . . . .	17
2.15	Программа в файле lab7-4.asm . . . . .	18
2.16	Запуск программы lab7-4.asm . . . . .	19

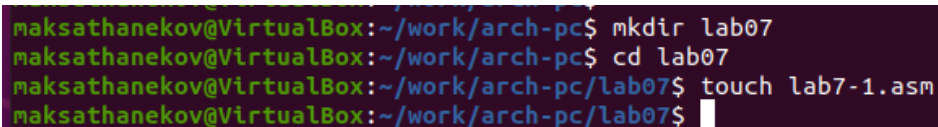
## Список таблиц

# 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

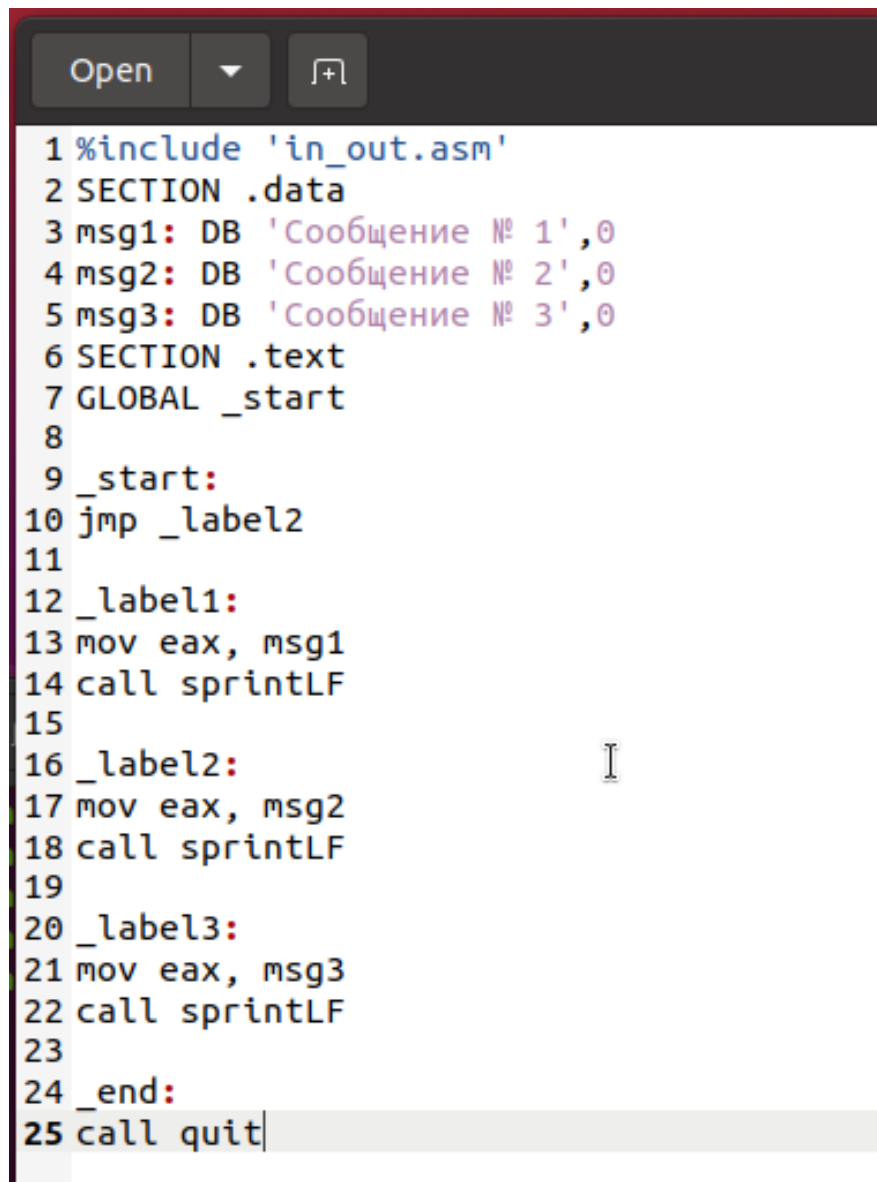
Создал каталог для программ лабораторной работы № 7 и файл lab7-1.asm.  
(рис. [2.1])

A screenshot of a terminal window with a dark purple background and light green text. It shows four lines of commands and their execution paths. The first line creates a directory 'lab07' in the home directory of the user 'maksathanekov'. The second line changes the current directory to 'lab07'. The third line creates a file 'lab7-1.asm' in the current directory. The fourth line shows the prompt after the file creation.

```
maksathanekov@VirtualBox:~/work/arch-pc$ mkdir lab07
maksathanekov@VirtualBox:~/work/arch-pc$ cd lab07
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ touch lab7-1.asm
maksathanekov@VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.1: Создал каталог и файл

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 в файле lab7-1.asm

Создал исполняемый файл и запустил его. (рис. [2.3])

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

Рис. 2.3: Запуск программы lab7-1.asm

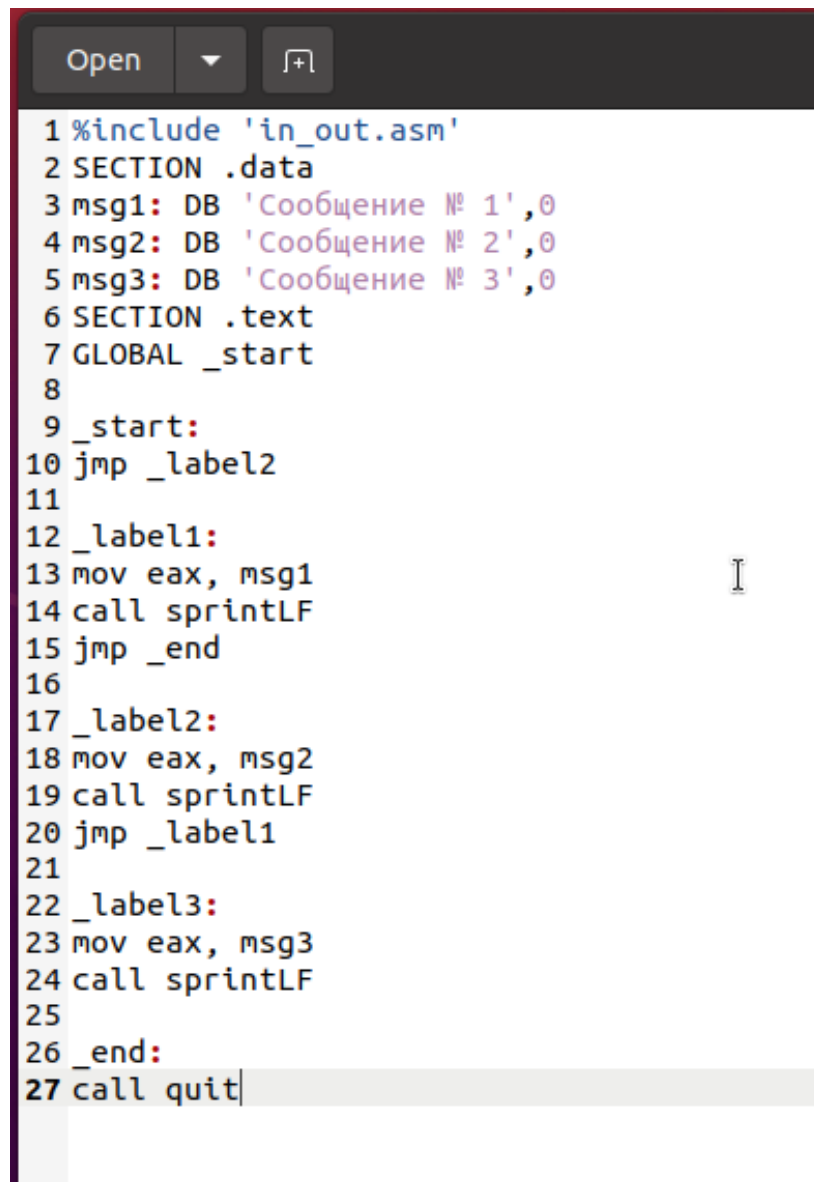
Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е.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

Изменил текст программы в соответствии с листингом 7.2. (рис. [2.4]) (рис. [2.5])

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

Рис. 2.4: Программа в файле 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
```

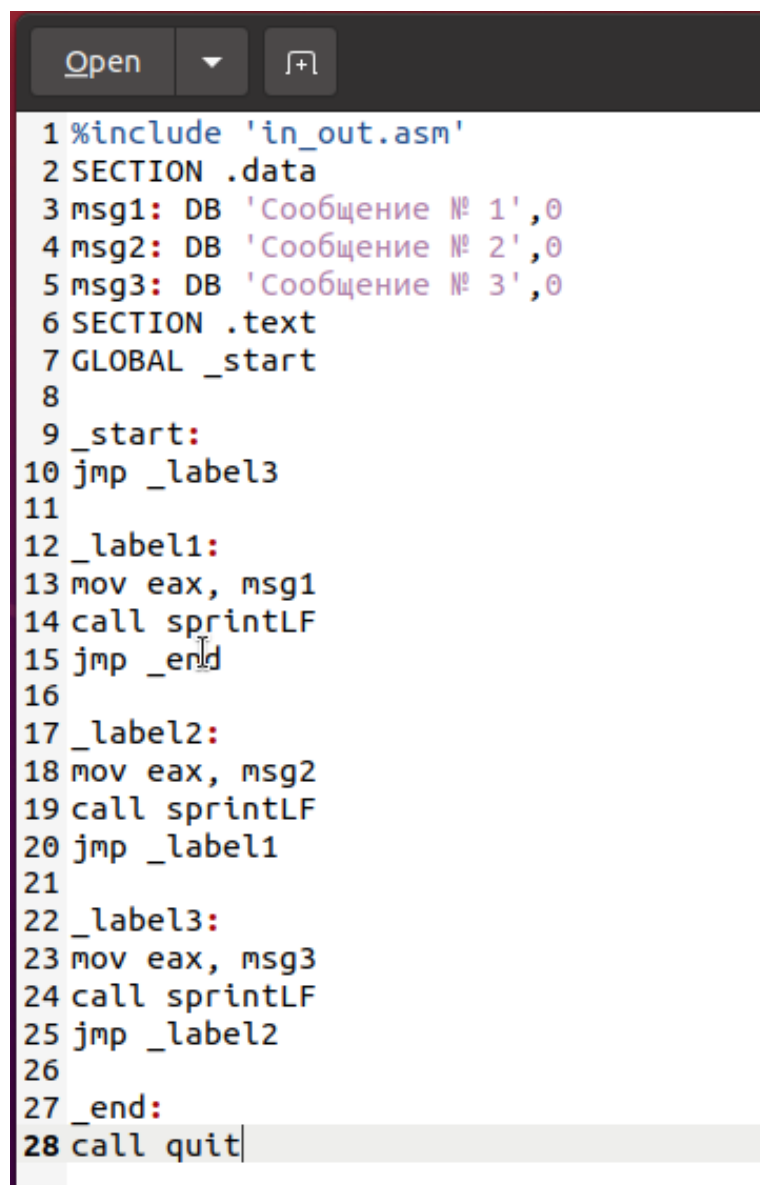
Рис. 2.5: Запуск программы lab7-1.asm

Изменил текст программы, изменив инструкции jmp, 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 в файле lab7-1.asm

```

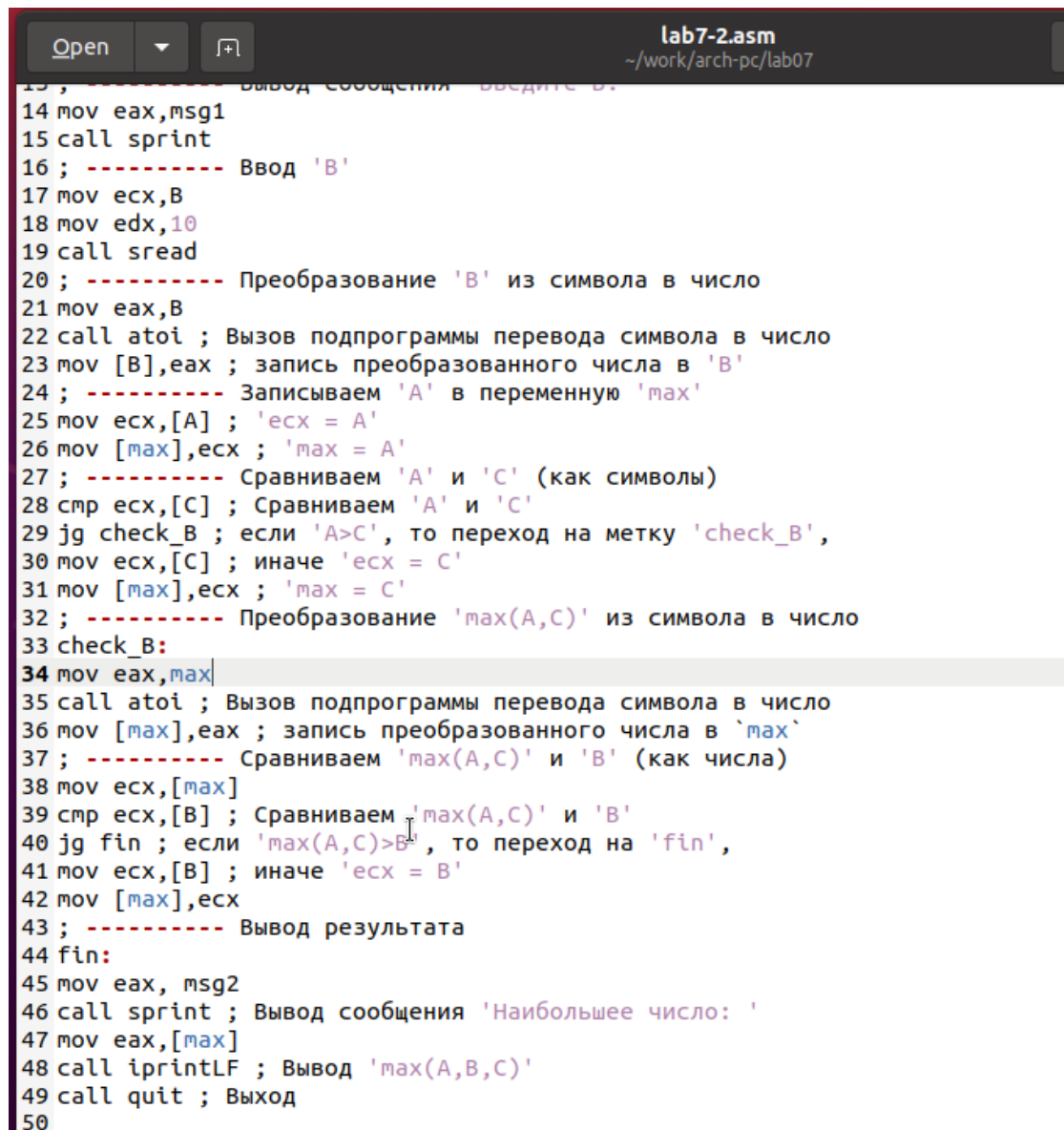
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
maksathanekov@VirtualBox:~/work/arch-pc/lab07$
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
maksathanekov@VirtualBox:~/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: A, B и C. Значения для A и C задаются в программе, значение B вводится с клавиатуры.

Создал исполняемый файл и проверил его работу для разных значений B (рис. [2.8]) (рис. [2.9]).



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

13 ; ----- Вывод сообщения "Введите В:"
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод 'В'
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'В' из символа в число
21 mov eax,B
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'В'
24 ; ----- Записываем 'А' в переменную 'max'
25 mov ecx,[A] ; 'ecx = A'
26 mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'А' и 'С' (как символы)
28 cmp ecx,[C] ; Сравниваем 'А' и 'С'
29 jg check_B ; если 'А>С', то переход на метку 'check_B',
30 mov ecx,[C] ; иначе 'ecx = C'
31 mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,max
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [max],eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'В' (как числа)
38 mov ecx,[max]
39 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'В'
40 jg fin ; если 'max(A,C)>В', то переход на 'fin',
41 mov ecx,[B] ; иначе 'ecx = В'
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 mov eax, msg2
46 call sprint ; Вывод сообщения 'Наибольшее число: '
47 mov eax,[max]
48 call iprintLF ; Вывод 'max(A,B,C)'
49 call quit ; Выход
50
```

Рис. 2.8: Программа в файле lab7-2.asm

```

maksathanekov@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 40
Наибольшее число: 50
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
maksathanekov@VirtualBox:~/work/arch-pc/lab07$
maksathanekov@VirtualBox:~/work/arch-pc/lab07$

```

Рис. 2.9: Запуск программы lab7-2.asm

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. [2.10])

```

lab7-2.asm
lab7-2.lst
179 4 00000025 00B5200187D0B8D181-
180 4 0000002E D0BBD0BE3A2000
181 5 00000035 32300000 A dd '20'
182 6 00000039 35300000 C dd '50'
183 7 section .bss
184 8 00000000 <res 0000000A> max resb 10
185 9 0000000A <res 0000000A> B resb 10
186 10 section .text
187 11 global _start
188 12 _start:
189 13 ; ----- Вывод сообщения 'Введите B: '
190 14 000000E8 B8[00000000] mov eax,msg1
191 15 000000ED E81DFFFFFF call sprint
192 16 ; ----- Ввод 'B'
193 17 000000F2 B9[0A000000] mov ecx,B
194 18 000000F7 BA0A000000 mov edx,10
195 19 000000FC E842FFFFFF call sread
196 20 ; ----- Преобразование 'B' из символа в число
197 21 00000101 B8[0A000000] mov eax,B
198 22 00000106 E891FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
199 23 0000010B A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
200 24 ; ----- Записываем 'A' в переменную 'max'
201 25 00000110 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
202 26 00000116 890D[00000000] mov [max],ecx ; 'max = A'
203 27 ; ----- Сравниваем 'A' и 'C' (как символы)
204 28 0000011C 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
205 29 00000122 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B',
206 30 00000124 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
207 31 0000012A 890D[00000000] mov [max],ecx ; 'max = C'
208 32 ; ----- Преобразование 'max(A,C)' из символа в число
209 33 check_B:
210 34 00000130 B8[00000000] mov eax,max
211 35 00000135 E862FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
212 36 0000013A A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
213 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
214 38 0000013F 8B0D[00000000] mov ecx,[max]

```

Рис. 2.10: Файл листинга lab7-2

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

строка 190

- 14 - номер строки в подпрограмме
- 000000E8 - адрес
- B8[00000000] - машинный код
- mov eax,msg1 - код программы - перекладывает msg1 в eax

строка 191

- 15 - номер строки в подпрограмме
- 000000ED - адрес
- E81DFFFFFF - машинный код
- call sprint - код программы - вызов подпрограммы печати

строка 193

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

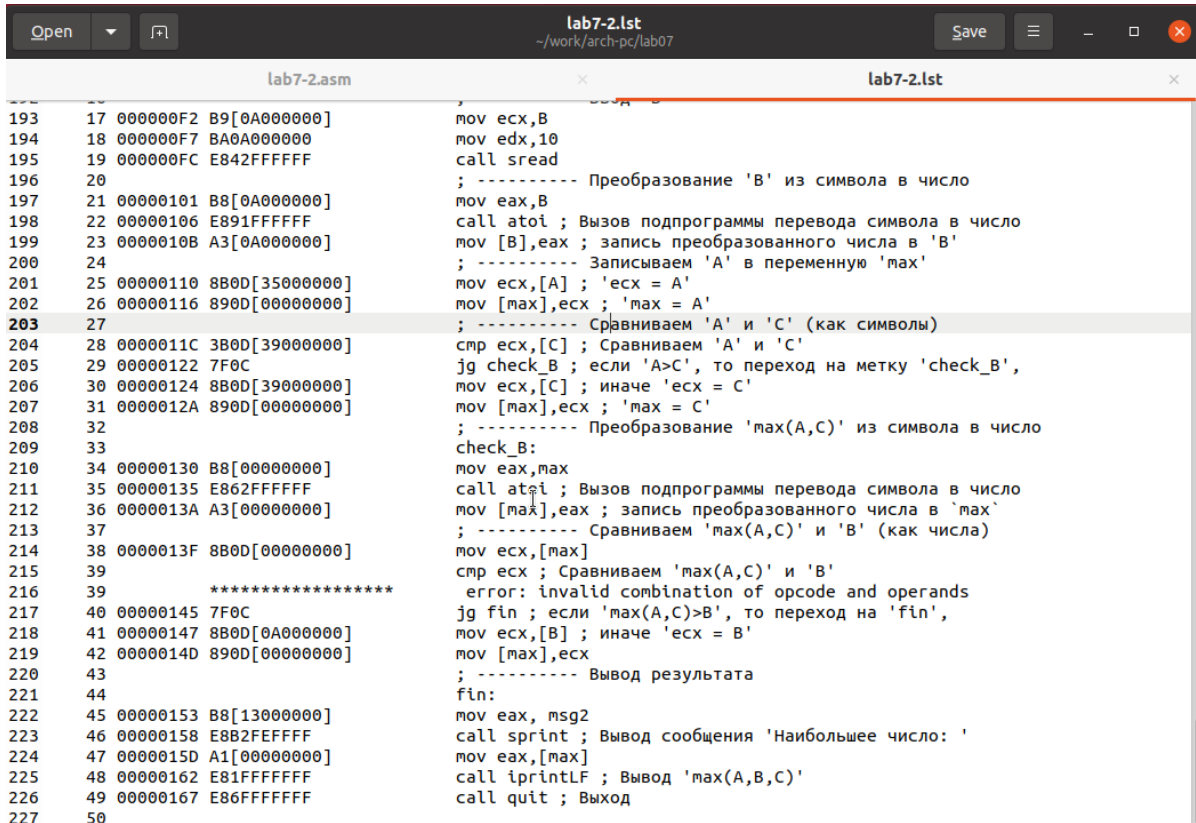
Открыл файл с программой lab7-2.asm и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга. (рис. [2.11]) (рис. [2.12])

```

maksathanekov@VirtualBox:~/work/arch-pc/lab07$
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.
lst
lab7-2.asm:39: error: invalid combination of opcode and operands
maksathanekov@VirtualBox:~/work/arch-pc/lab07$
maksathanekov@VirtualBox:~/work/arch-pc/lab07$

```

Рис. 2.11: Ошибка трансляции lab7-2



```

lab7-2.asm
lab7-2.lst
193 17 000000F2 B9[0A000000]    mov ecx,B
194 18 000000F7 BA0A000000    mov edx,10
195 19 000000FC E842FFFFFF    call sread
196 20
197 21 00000101 B8[0A000000]    mov eax,B
198 22 00000106 E891FFFFFF    call atoi ; Вызов подпрограммы перевода символа в число
199 23 0000010B A3[0A000000]    mov [B],eax ; запись преобразованного числа в 'B'
200 24
201 25 00000110 8B0D[35000000]    mov ecx,[A] ; 'ecx = A'
202 26 00000116 890D[00000000]    mov [max],ecx ; 'max = A'
203 27
204 28 0000011C 3B0D[39000000]    cmp ecx,[C] ; Сравниваем 'A' и 'C' (как символы)
205 29 00000122 7F0C    jg check_B ; если 'A>C', то переход на метку 'check_B',
206 30 00000124 8B0D[39000000]    mov ecx,[C] ; иначе 'ecx = C'
207 31 0000012A 890D[00000000]    mov [max],ecx ; 'max = C'
208 32
209 33
210 34 00000130 B8[00000000]    mov eax,max
211 35 00000135 E862FFFFFF    call atoi ; Вызов подпрограммы перевода символа в число
212 36 0000013A A3[00000000]    mov [max],eax ; запись преобразованного числа в 'max'
213 37
214 38 0000013F 8B0D[00000000]    mov ecx,[max]
215 39
216 39 *****
217 40 00000145 7F0C    jg fin ; если 'max(A,C)>B', то переход на 'fin',
218 41 00000147 8B0D[0A000000]    mov ecx,[B] ; иначе 'ecx = B'
219 42 0000014D 890D[00000000]    mov [max],ecx
220 43
221 44
222 45 00000153 B8[13000000]    mov eax,msg2
223 46 00000158 E8B2FFFFFF    call sprintf ; Вывод сообщения 'Наибольшее число: '
224 47 0000015D A1[00000000]    mov eax,[max]
225 48 00000162 E81FFFFFFF    call iprintf ; Вывод 'max(A,B,C)'
226 49 00000167 E86FFFFFFF    call quit ; Выход
227 50

```

Рис. 2.12: Файл листинга с ошибкой lab7-2

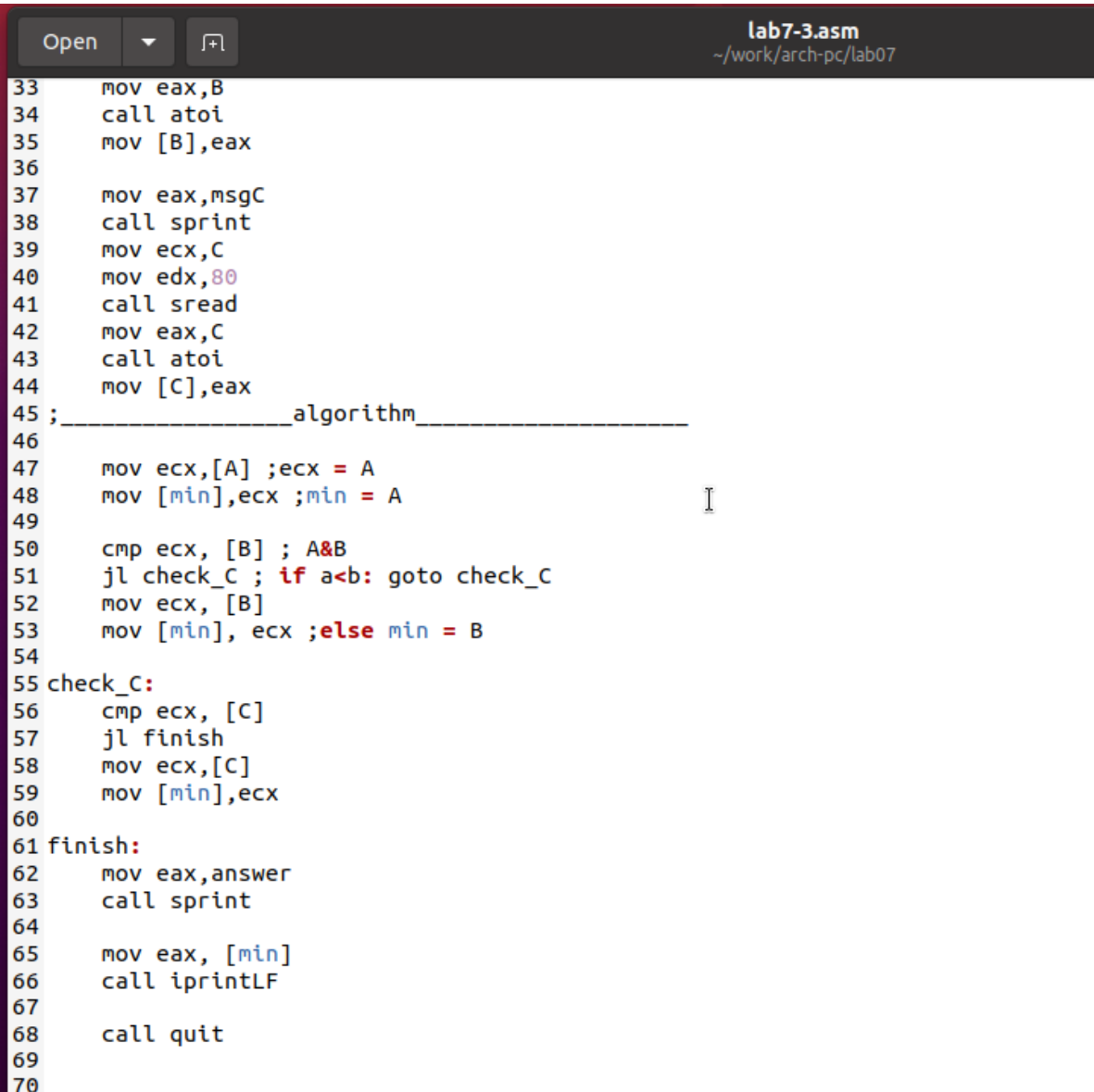
Объектный файл не смог создаться из-за ошибки. Но получился листинг, где выделено место ошибки.

## 2.1 Самостоятельное задание

Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариан-

том, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу (рис. [2.13]) (рис. [2.14])

для варианта 13 - 84,32,77



```
lab7-3.asm
~/work/arch-pc/lab07

33  mov eax,B
34  call atoi
35  mov [B],eax
36
37  mov eax,msgC
38  call sprintf
39  mov ecx,C
40  mov edx,80
41  call sread
42  mov eax,C
43  call atoi
44  mov [C],eax
45 ;_____algorithm_____
46
47  mov ecx,[A] ;ecx = A
48  mov [min],ecx ;min = A
49
50  cmp ecx, [B] ; A&B
51  jl check_C ; if a<b: goto check_C
52  mov ecx, [B]
53  mov [min], ecx ;else min = B
54
55 check_C:
56  cmp ecx, [C]
57  jl finish
58  mov ecx,[C]
59  mov [min],ecx
60
61 finish:
62  mov eax,answer
63  call sprintf
64
65  mov eax, [min]
66  call iprintLF
67
68  call quit
69
70
```

Рис. 2.13: Программа в файле lab7-3.asm



```

maksathanekov@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Input A: 84
Input B: 32
Input C: 77
Smallest: 32
maksathanekov@VirtualBox:~/work/arch-pc/lab07$
maksathanekov@VirtualBox:~/work/arch-pc/lab07$

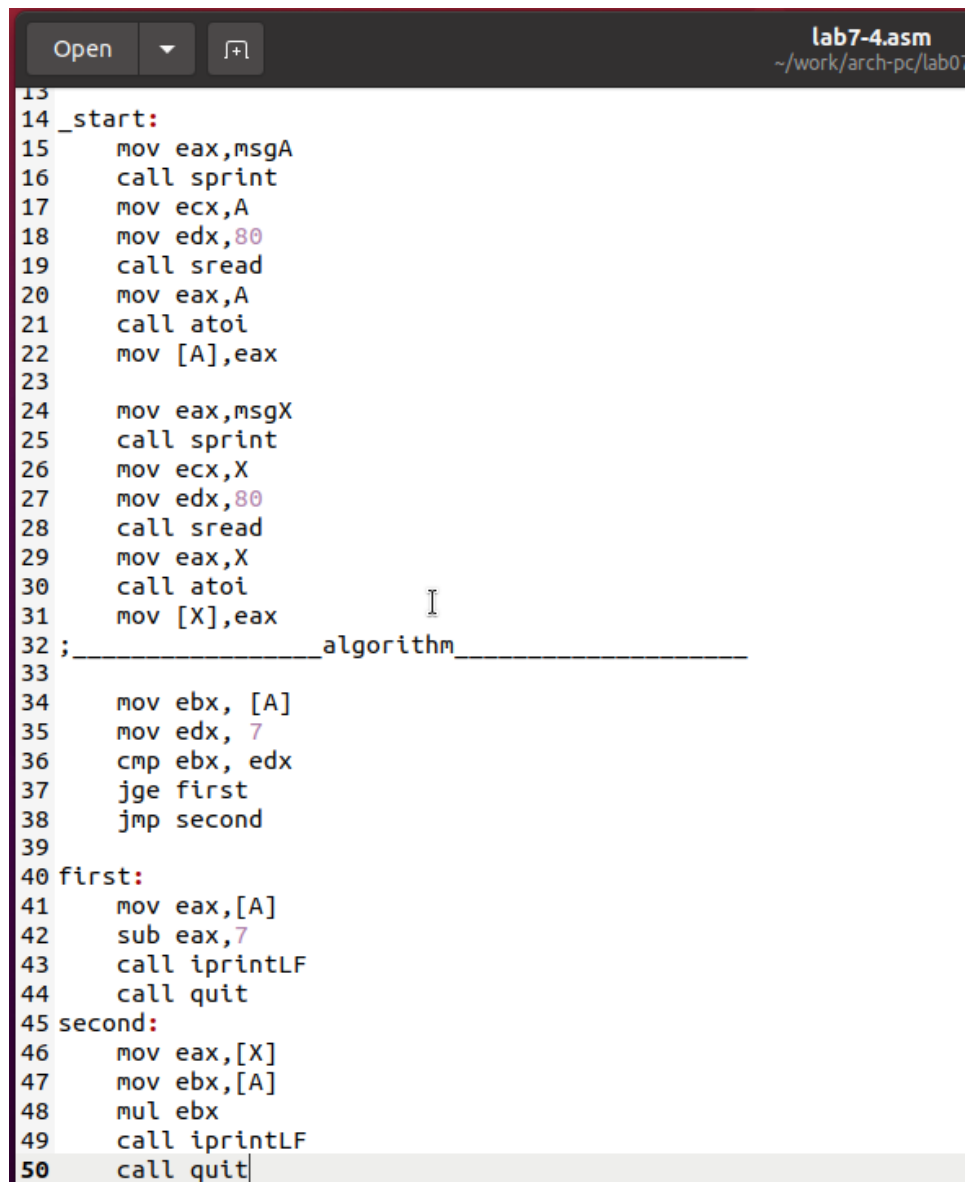
```

Рис. 2.14: Запуск программы lab7-3.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 13

$$\begin{cases} a - 7, a \geq 7 \\ ax, a < 7 \end{cases}$$



```
lab7-4.asm
~/work/arch-pc/lab07

13
14 _start:
15     mov eax,msgA
16     call sprintf
17     mov ecx,A
18     mov edx,80
19     call sread
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 ;_____algorithm_____
33
34     mov ebx, [A]
35     mov edx, 7
36     cmp ebx, edx
37     jge first
38     jmp second
39
40 first:
41     mov eax,[A]
42     sub eax,7
43     call iprintLF
44     call quit
45 second:
46     mov eax,[X]
47     mov ebx,[A]
48     mul ebx
49     call iprintLF
50     call quit
```

Рис. 2.15: Программа в файле lab7-4.asm

```
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ ./lab7-4
Input A: 9
Input X: 3
2
maksathanekov@VirtualBox:~/work/arch-pc/lab07$ ./lab7-4
Input A: 4
Input X: 6
24
maksathanekov@VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы lab7-4.asm

## 3 Выводы

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