

# **Отчёт по лабораторной работе 7**

**Архитектура компьютеров**

Тефера Небию Десси

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Выводы</b>	<b>20</b>

## Список иллюстраций

2.1	Программа в файле lab7-1.asm . . . . .	7
2.2	Запуск программы lab7-1.asm . . . . .	8
2.3	Программа в файле lab7-1.asm: . . . . .	9
2.4	Запуск программы lab7-1.asm: . . . . .	9
2.5	Программа в файле lab7-1.asm . . . . .	10
2.6	Запуск программы lab7-1.asm . . . . .	11
2.7	Программа в файле lab7-2.asm . . . . .	12
2.8	Запуск программы lab7-2.asm . . . . .	13
2.9	Файл листинга lab7-2 . . . . .	13
2.10	Ошибка трансляции lab7-2 . . . . .	15
2.11	Файл листинга с ошибкой lab7-2 . . . . .	15
2.12	Программа в файле task.asm . . . . .	16
2.13	Запуск программы task.asm . . . . .	17
2.14	Программа в файле task2.asm . . . . .	18
2.15	Запуск программы task2.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

1. Создал каталог для программ лабораторной работы № 7 и файл lab7-1.asm
2. 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.

```
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 sprintfLF
15
16 _label2:
17 mov eax, msg2
18 call sprintfLF
19
20 _label3:
21 mov eax, msg3
22 call sprintfLF
23
24 _end:
25 call quit
```

Рис. 2.1: Программа в файле lab7-1.asm

Создал исполняемый файл и запустил его.

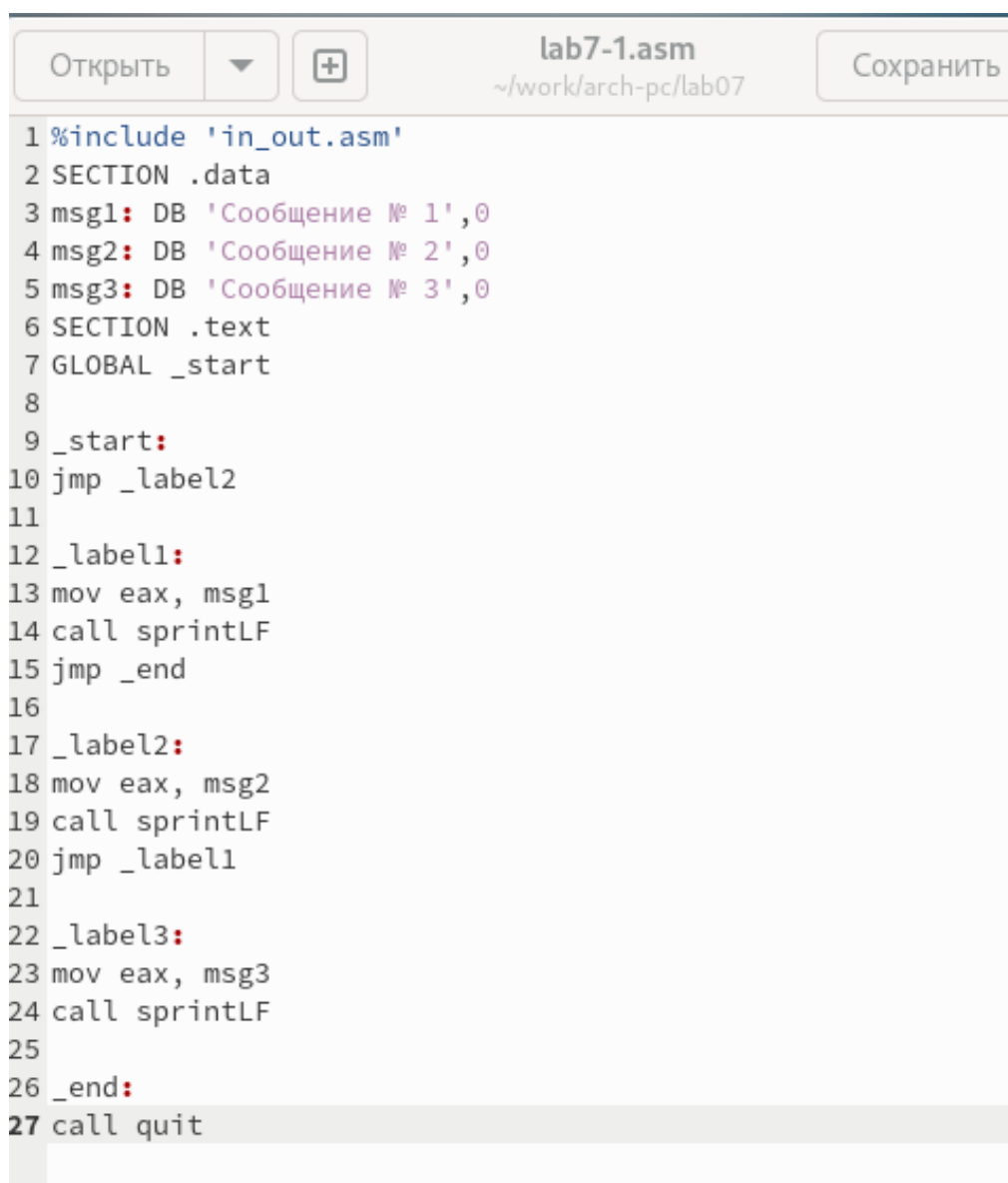
```
nebiyudessie@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
nebiyudessie@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
nebiyudessie@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
nebiyudessie@vbox:~/work/arch-pc/lab07$ █
```

Рис. 2.2: 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.

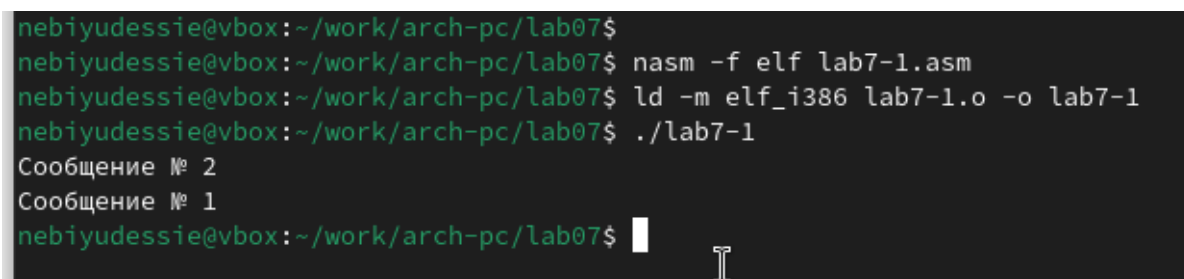




```
lab7-1.asm
~/work/arch-pc/lab07

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. 2.3: Программа в файле lab7-1.asm:



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

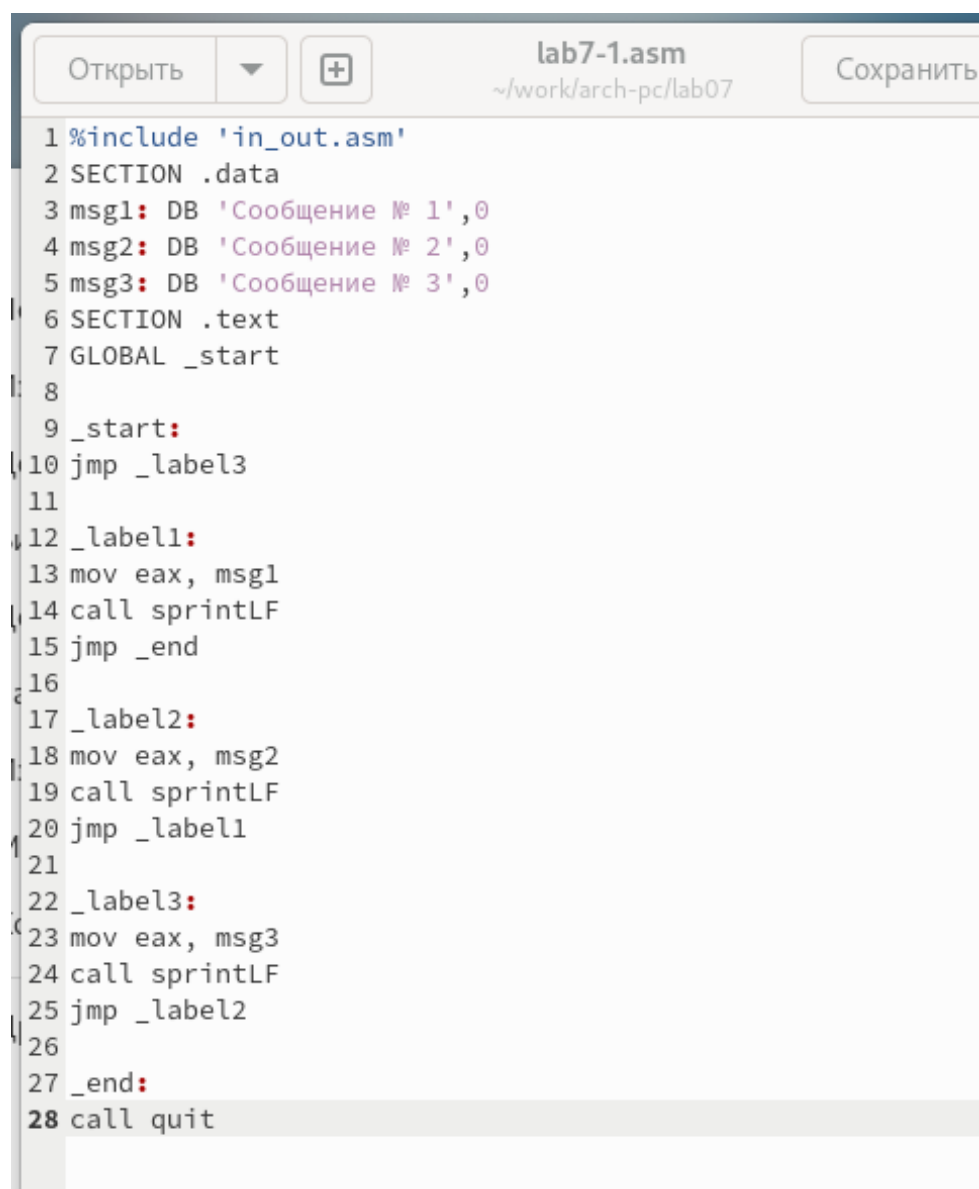
Рис. 2.4: Запуск программы lab7-1.asm:

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

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. 2.5: Программа в файле lab7-1.asm

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

Рис. 2.6: Запуск программы lab7-1.asm

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: 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.

```
17 mov ecx,B
18 mov edx,10
19 call sread
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. 2.7: Программа в файле lab7-2.asm

```

nebiyudessie@vbox:~/work/arch-pc/lab07$
nebiyudessie@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
nebiyudessie@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
nebiyudessie@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 30
Наибольшее число: 50
nebiyudessie@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 55
Наибольшее число: 55
nebiyudessie@vbox:~/work/arch-pc/lab07$

```

Рис. 2.8: Запуск программы lab7-2.asm

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.

Создал файл листинга для программы из файла lab7-2.asm

```

189 14 000000E8 B8[00000000]    mov eax,msg1
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 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

```

Рис. 2.9: Файл листинга lab7-2

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

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

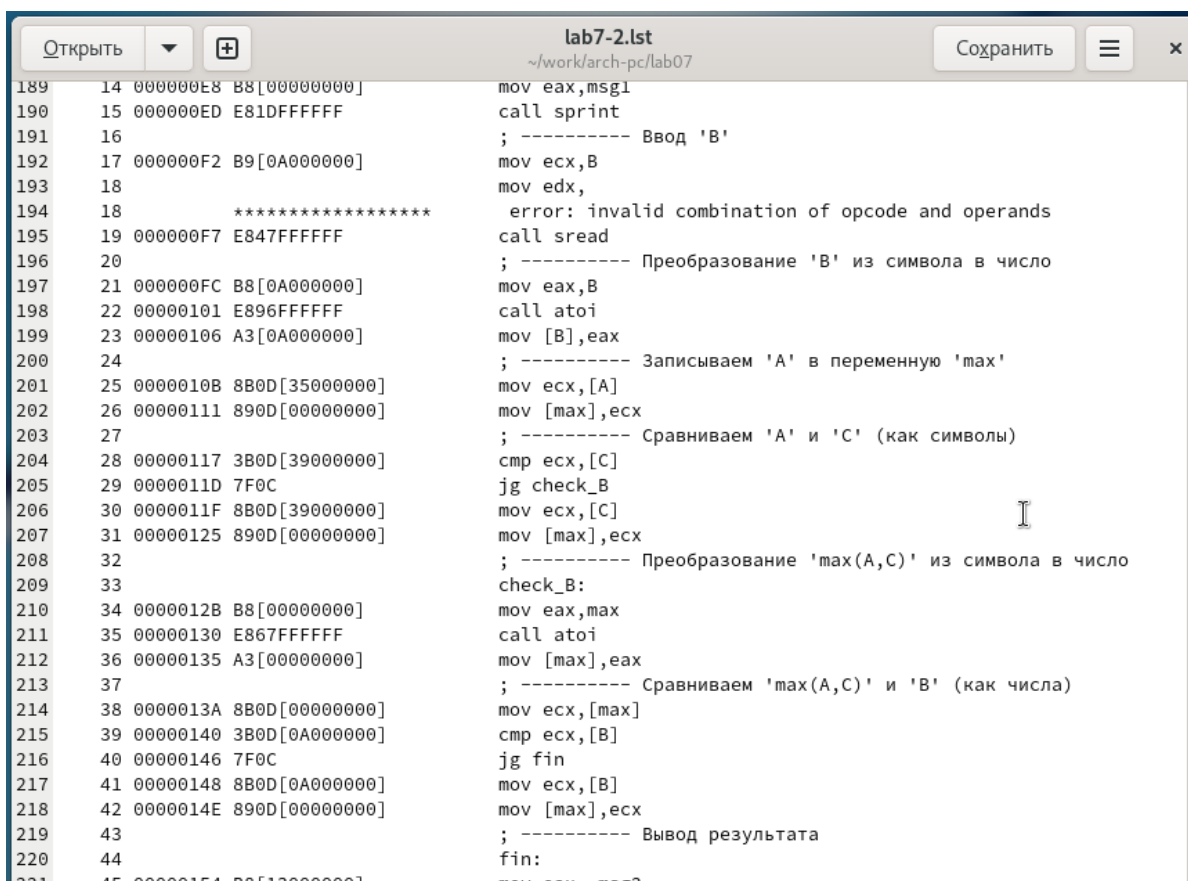
строка 213

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

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

```
nebiyudessie@vbox:~/work/arch-pc/lab07$
nebiyudessie@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
nebiyudessie@vbox:~/work/arch-pc/lab07$
nebiyudessie@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:18: error: invalid combination of opcode and operands
nebiyudessie@vbox:~/work/arch-pc/lab07$
nebiyudessie@vbox:~/work/arch-pc/lab07$
```

Рис. 2.10: Ошибка трансляции lab7-2



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

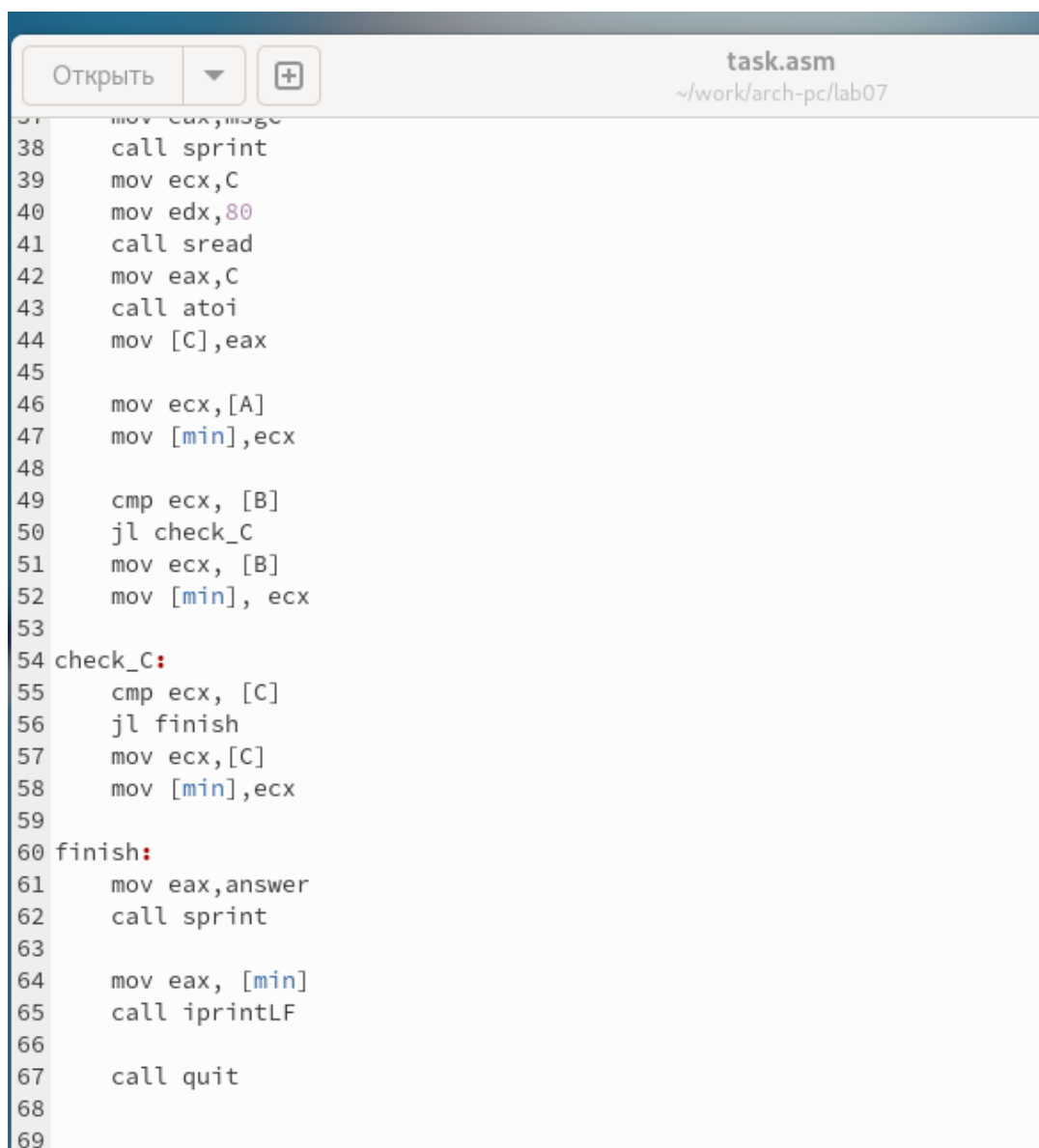
189 14 000000E8 B8[00000000] mov eax,msg1
190 15 000000ED E81DFFFFFF call sprint
191 16 ; ----- Ввод 'B'
192 17 000000F2 B9[0A000000] mov ecx,B
193 18 mov edx,
194 18 *****
195 19 000000F7 E847FFFFFF call sread
196 20 ; ----- Преобразование 'B' из символа в число
197 21 000000FC B8[0A000000] mov eax,B
198 22 00000101 E896FFFFFF call atoi
199 23 00000106 A3[0A000000] mov [B],eax
200 24 ; ----- Записываем 'A' в переменную 'max'
201 25 0000010B 8B0D[35000000] mov ecx,[A]
202 26 00000111 890D[00000000] mov [max],ecx
203 27 ; ----- Сравниваем 'A' и 'C' (как символы)
204 28 00000117 3B0D[39000000] cmp ecx,[C]
205 29 0000011D 7F0C jg check_B
206 30 0000011F 8B0D[39000000] mov ecx,[C]
207 31 00000125 890D[00000000] mov [max],ecx
208 32 ; ----- Преобразование 'max(A,C)' из символа в число
209 33 check_B:
210 34 0000012B B8[00000000] mov eax,max
211 35 00000130 E867FFFFFF call atoi
212 36 00000135 A3[00000000] mov [max],eax
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[12000000] mov eax,msg2
```

Рис. 2.11: Файл листинга с ошибкой lab7-2

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

5. 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



```
37  mov ecx, message
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:
61  mov eax, answer
62  call sprint
63
64  mov eax, [min]
65  call iprintLF
66
67  call quit
68
69
```

Рис. 2.12: Программа в файле task.asm



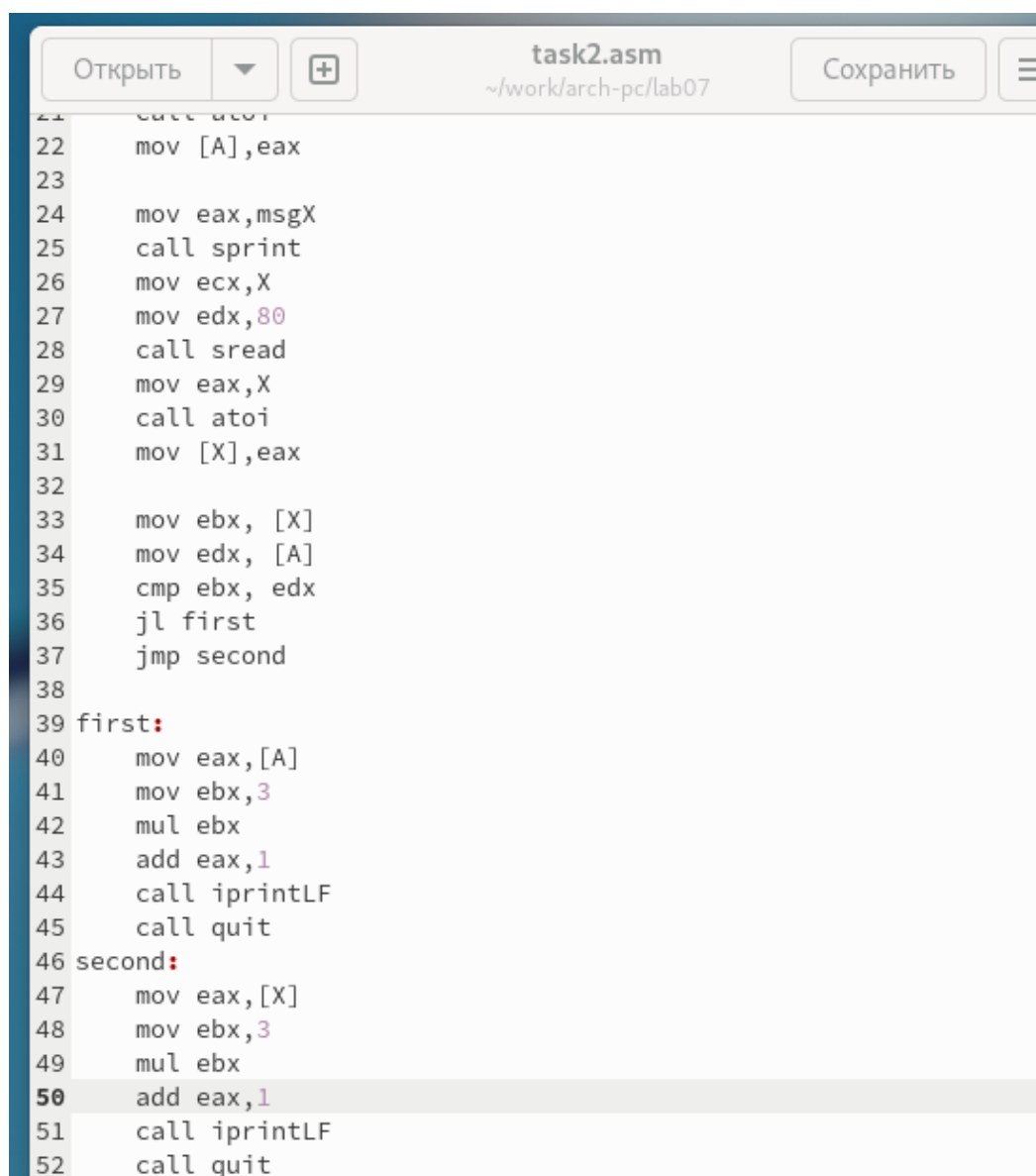
```
nebiyudessie@vbox:~/work/arch-pc/lab07$  
nebiyudessie@vbox:~/work/arch-pc/lab07$ nasm -f elf task.asm  
nebiyudessie@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 task.o -o task  
nebiyudessie@vbox:~/work/arch-pc/lab07$ ./task  
Input A: 81  
Input B: 22  
Input C: 72  
Smallest: 22  
nebiyudessie@vbox:~/work/arch-pc/lab07$
```

Рис. 2.13: Запуск программы task.asm

6. 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

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



```
21     call atoi
22     mov [A],eax
23
24     mov eax,msgX
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 ebx, [X]
34     mov edx, [A]
35     cmp ebx, edx
36     jl 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. 2.14: Программа в файле task2.asm

```
nebiyudessie@vbox:~/work/arch-pc/lab07$  
nebiyudessie@vbox:~/work/arch-pc/lab07$ nasm -f elf task2.asm  
nebiyudessie@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 task2.o -o task2  
nebiyudessie@vbox:~/work/arch-pc/lab07$ ./task2  
Input A: 3  
Input X: 2  
10  
nebiyudessie@vbox:~/work/arch-pc/lab07$ ./task2  
Input A: 2  
Input X: 4  
13  
nebiyudessie@vbox:~/work/arch-pc/lab07$
```

Рис. 2.15: Запуск программы task2.asm

## 3 Выводы

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