

Отчёт по лабораторной работе 7

дисциплина: Архитектура компьютера

Симдянов Тимур НПИбд-03-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файла листинга	12
2.3	Самостоятельное задание	14
3	Выводы	19

Список иллюстраций

2.1	Программа lab7-1.asm	7
2.2	Запуск программы lab7-1.asm	7
2.3	Программа lab7-1.asm	8
2.4	Запуск программы lab7-1.asm	9
2.5	Программа lab7-1.asm	10
2.6	Запуск программы lab7-1.asm	10
2.7	Программа lab7-2.asm	11
2.8	Запуск программы lab7-2.asm	12
2.9	Файл листинга lab7-2	12
2.10	Ошибка трансляции lab7-2	13
2.11	Файл листинга с ошибкой lab7-2	14
2.12	Программа lab7-3.asm	15
2.13	Запуск программы lab7-3.asm	15
2.14	Программа lab7-4.asm	17
2.15	Запуск программы lab7-4.asm	18

Список таблиц

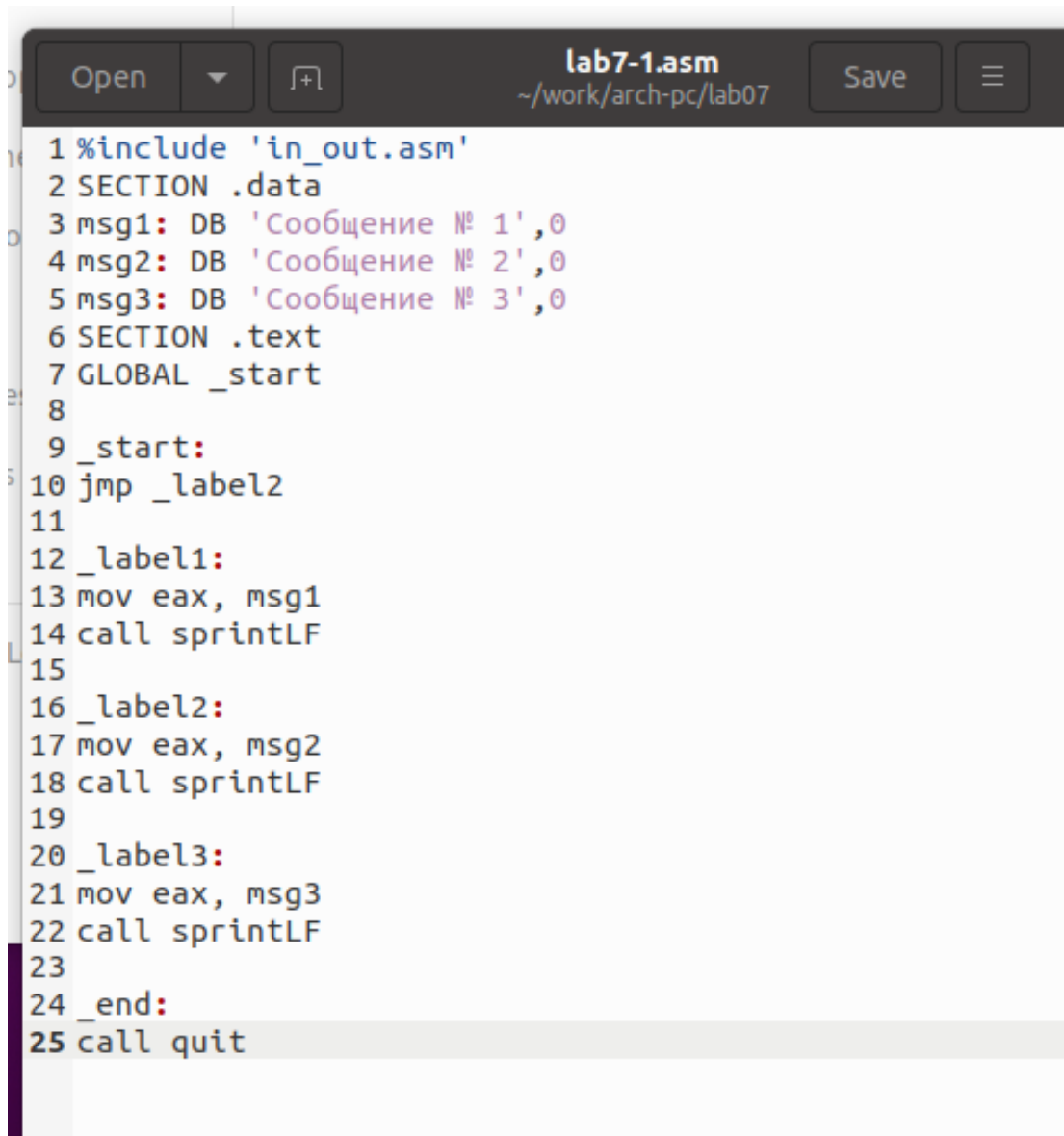
1 Цель работы

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

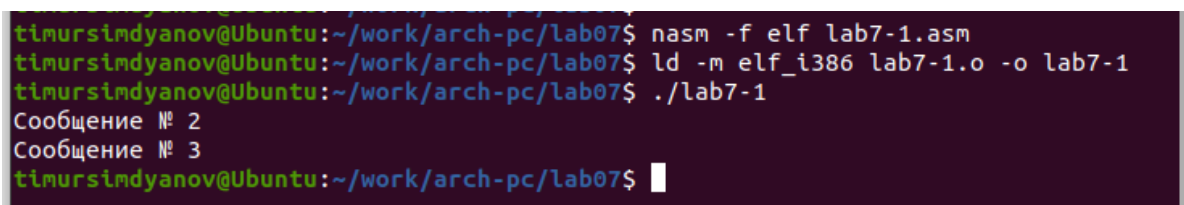
Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Написал в файл `lab7-1.asm` текст программы из листинга 7.1. (рис. 2.1)



```
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 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. 2.1: Программа lab7-1.asm

Создаю исполняемый файл и запускаю его. (рис. 2.2)

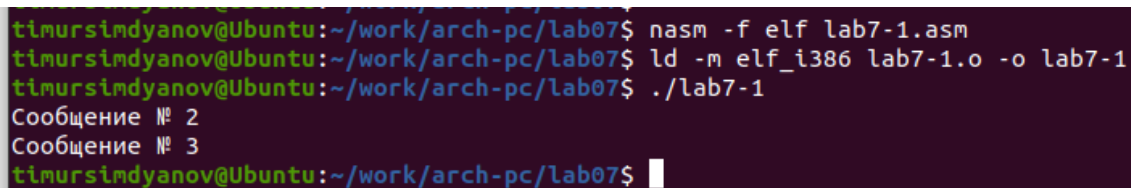


```
timursindyanov@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
timursindyanov@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
timursindyanov@Ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
timursindyanov@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.2: Запуск программы lab7-1.asm

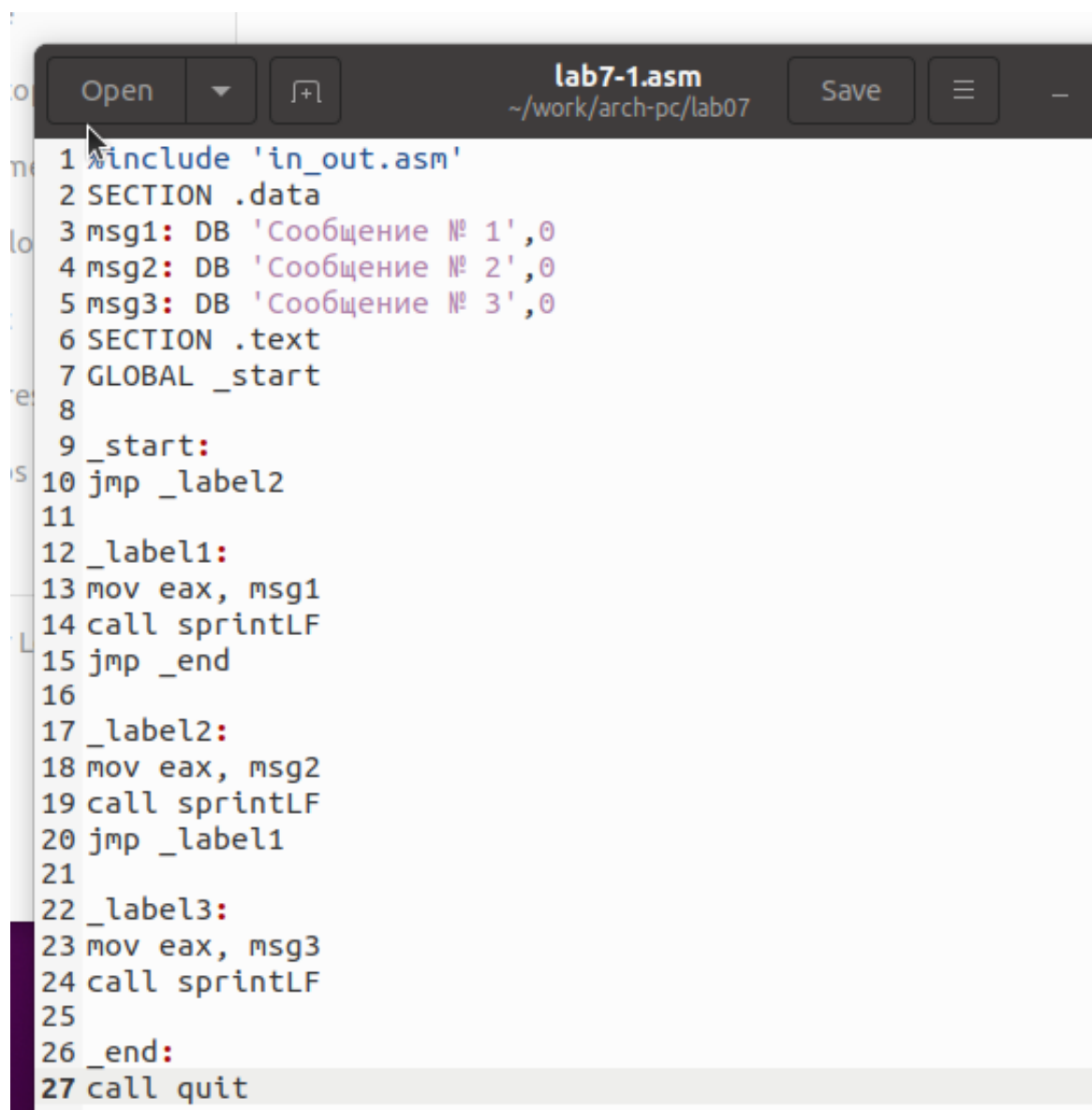
Инструкция `jmp` позволяет осуществлять переходы не только вперед, но и назад. Изменим программу так, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавляем инструкцию `jmp` с меткой `_label1` (т.е. переход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

Изменяю текст программы в соответствии с листингом 7.2. (рис. 2.3) (рис. 2.4)



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

Рис. 2.3: Программа lab7-1.asm

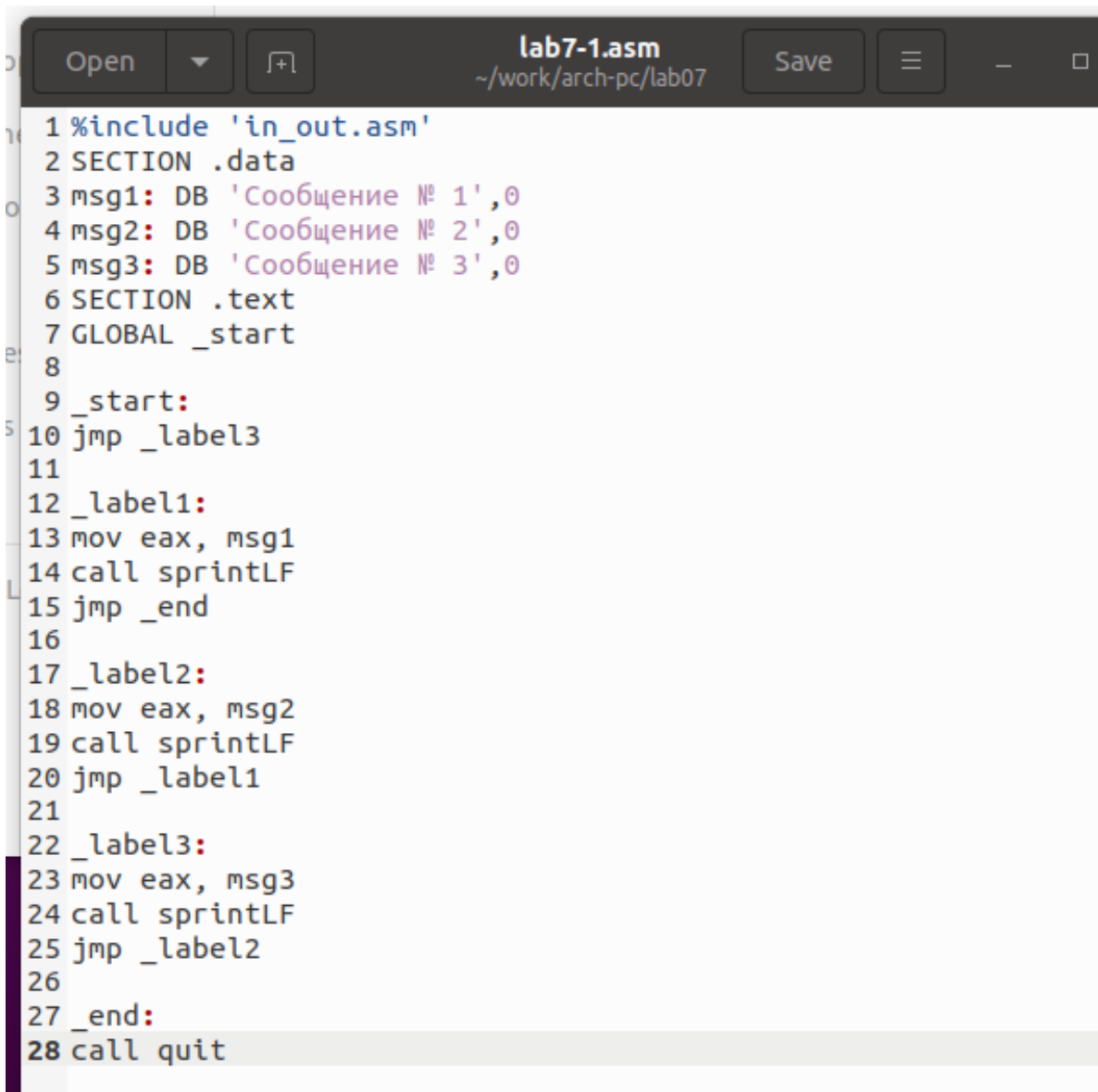


```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8
9 _start:
10 jmp _label2
11
12 _label1:
13 mov eax, msg1
14 call sprintLF
15 jmp _end
16
17 _label2:
18 mov eax, msg2
19 call sprintLF
20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

Рис. 2.4: Запуск программы lab7-1.asm

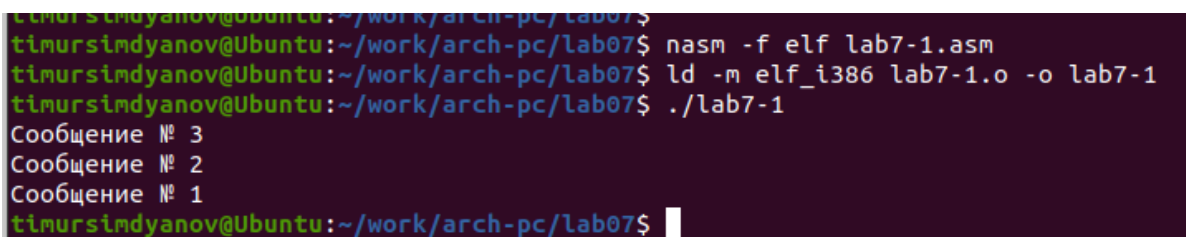
Изменил текст программы, чтобы вывод программы был следующим (рис. 2.5) (рис. 2.6):

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 lab7-1.asm



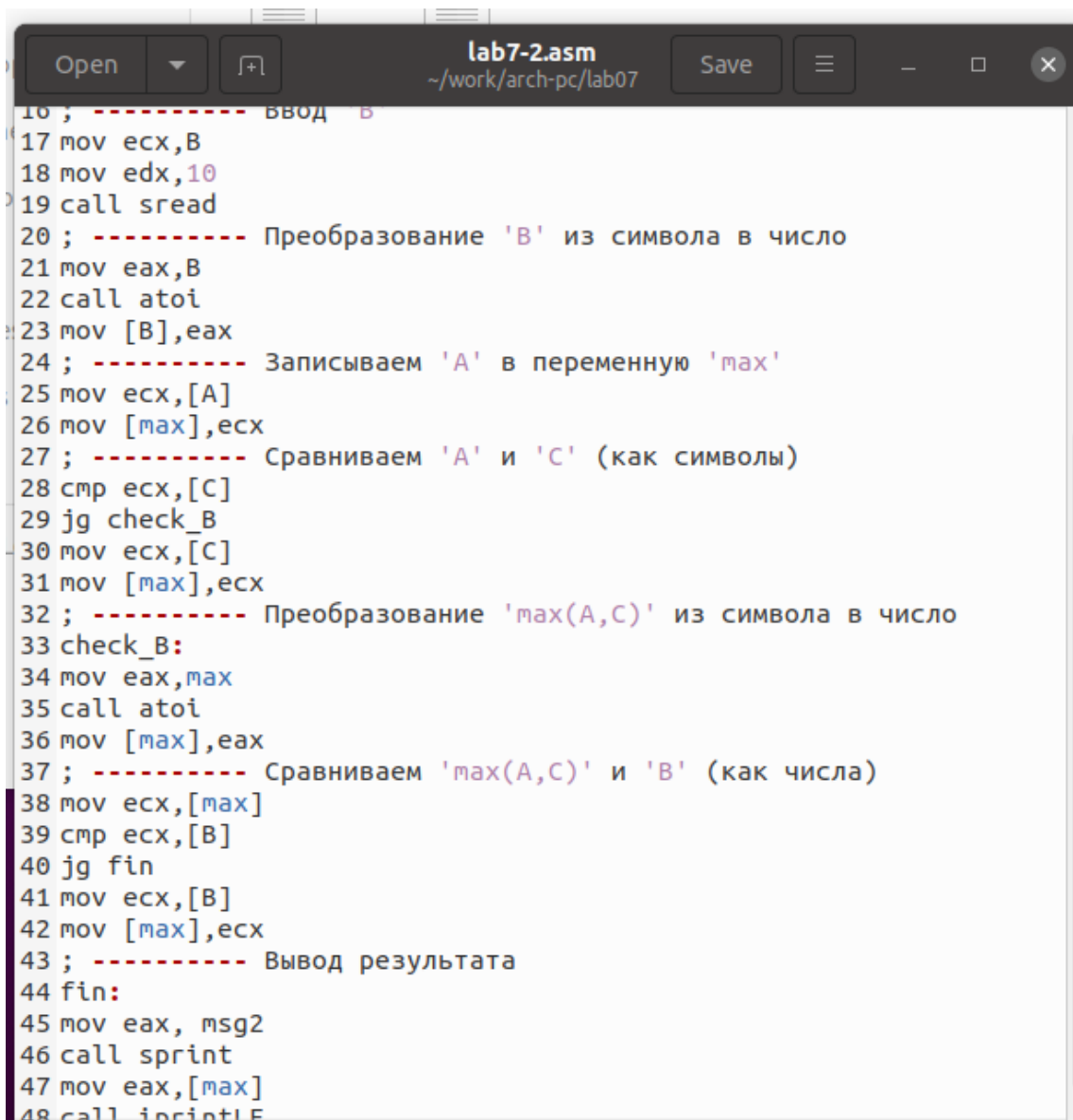
```
timursindyanov@ubuntu:~/work/arch-pc/lab07$
timursindyanov@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
timursindyanov@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
timursindyanov@ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
timursindyanov@ubuntu:~/work/arch-pc/lab07$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

Рис. 2.6: 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: A, B и C. Значения для A и C задаются в программе, значение B вводится с клавиатуры.

Создаю исполняемый файл и проверяю его работу для разных значений B (рис. 2.7) (рис. 2.8).



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

16 ; ----- Ввод 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 incintLE
```

Рис. 2.7: Программа lab7-2.asm

```

timursindyanov@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
timursindyanov@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
timursindyanov@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 30
Наибольшее число: 50
timursindyanov@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
timursindyanov@ubuntu:~/work/arch-pc/lab07$

```

Рис. 2.8: Запуск программы lab7-2.asm

2.2 Изучение структуры файла листинга

Обычно nasm создает 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. 2.9)

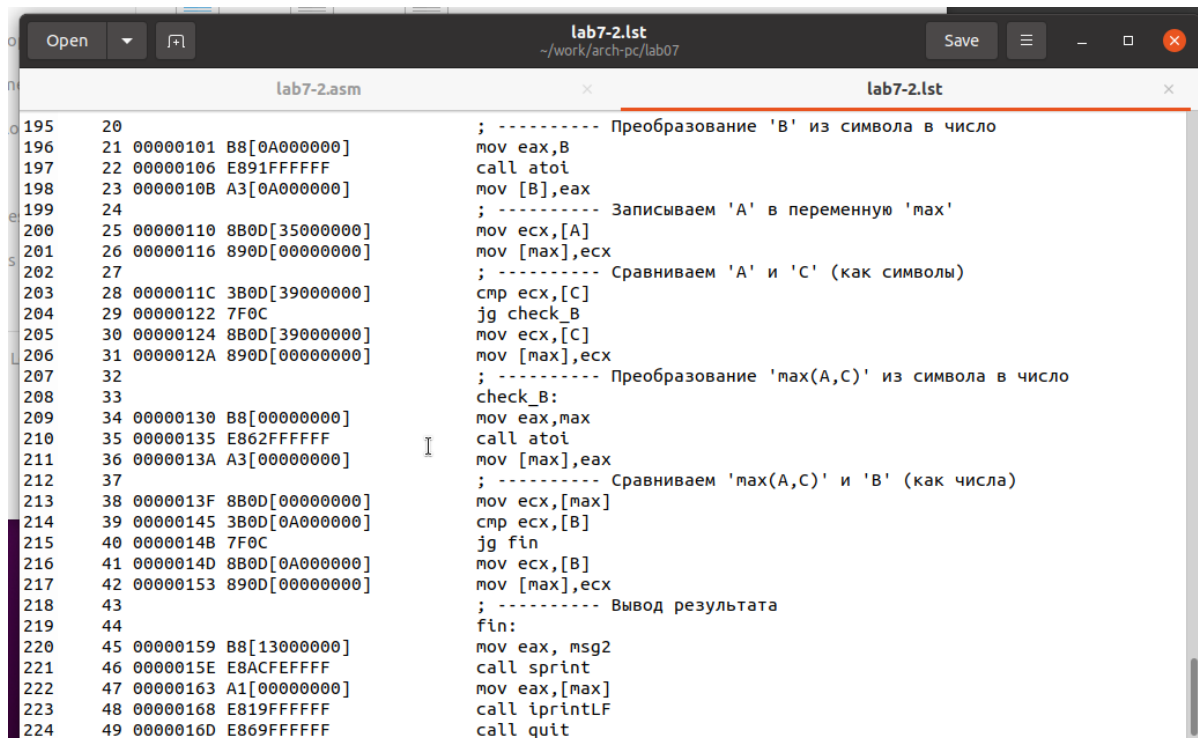


Рис. 2.9: Файл листинга lab7-2

Ознакомимся с его форматом и содержимым.

строка 203

- 28 - номер строки
- 0000011C - адрес
- 3B0D[39000000] - машинный код
- `str esx,[C]` - код программы

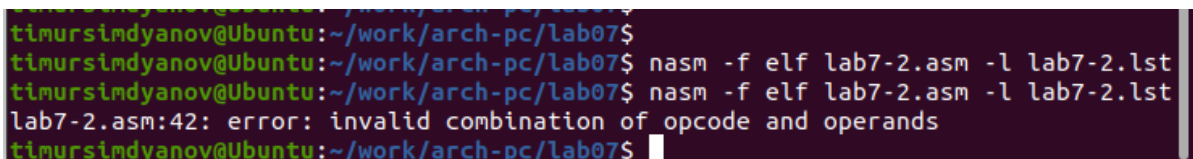
строка 204

- 29 - номер строки
- 00000122 - адрес
- 7F0C - машинный код
- `jg check_B` - код программы

строка 205

- 30 - номер строки
- 00000124 - адрес
- 8B0D[39000000] - машинный код
- `mov esx,[C]` - код программы

Открываю файл с программой `lab7-2.asm` и в инструкции с двумя операндами удаляю один операнд. Выполняю трансляцию с получением файла листинга. (рис. 2.10) (рис. 2.11)



```
timursimdyanov@Ubuntu:~/work/arch-pc/lab07$  
timursimdyanov@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
timursimdyanov@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
lab7-2.asm:42: error: invalid combination of opcode and operands  
timursimdyanov@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.10: Ошибка трансляции `lab7-2`

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

lab7-2.asm
lab7-2.lst

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 7F06                                jg fin
216 41 0000014D 8B0D[0A000000]    mov ecx,[B]
217 42                                mov [max],
218 42                                *****
219 43                                error: invalid combination of opcode and operands
220 44                                ; ----- Вывод результата
221 45 00000153 B8[13000000]    mov eax, msg2
222 46 00000158 E8B2FEFFFF    call sprint
223 47 0000015D A1[00000000]    mov eax,[max]
224 48 00000162 E81FFFFFFF    call iprintLF
225 49 00000167 E86FFFFFFF    call quit
```

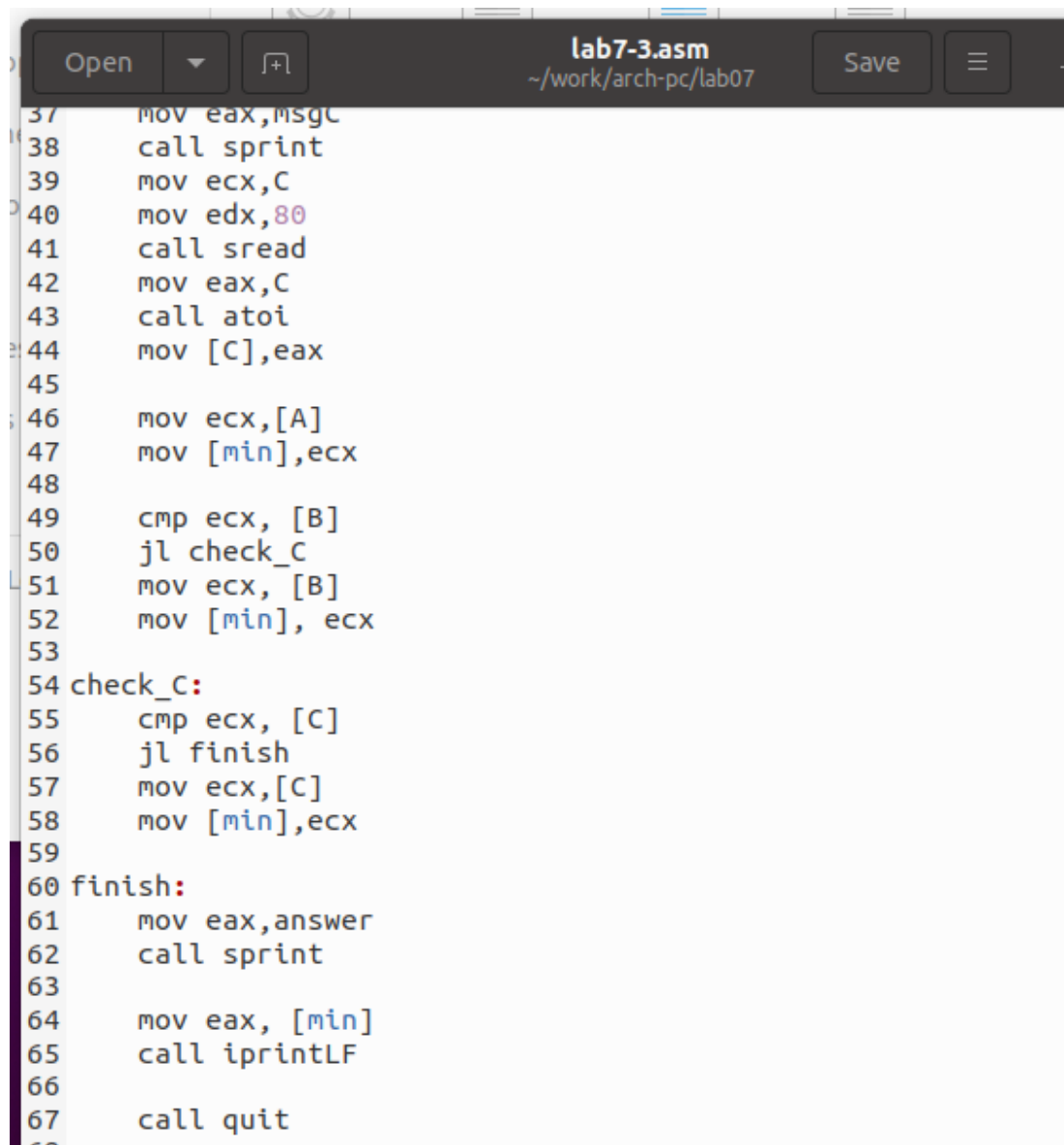
Рис. 2.11: Файл листинга с ошибкой lab7-2

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

2.3 Самостоятельное задание

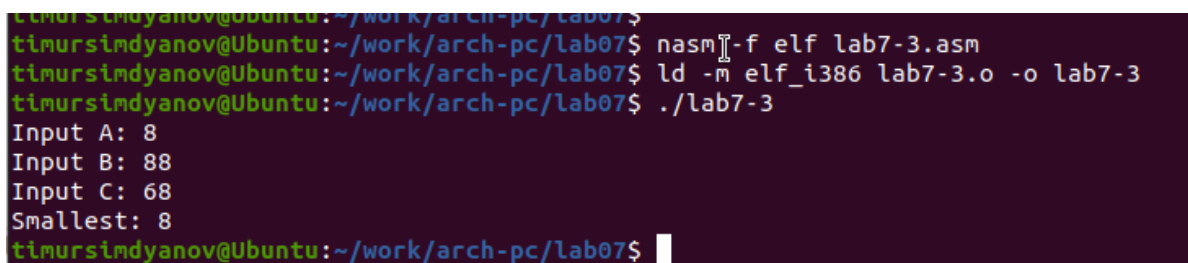
Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу (рис. 2.12) (рис. 2.13)

для варианта 4 - 8,88,68



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

Рис. 2.12: Программа lab7-3.asm



```
timursimdyanov@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
timursimdyanov@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
timursimdyanov@Ubuntu:~/work/arch-pc/lab07$ ./lab7-3
Input A: 8
Input B: 88
Input C: 68
Smallest: 8
timursimdyanov@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.13: Запуск программы lab7-3.asm

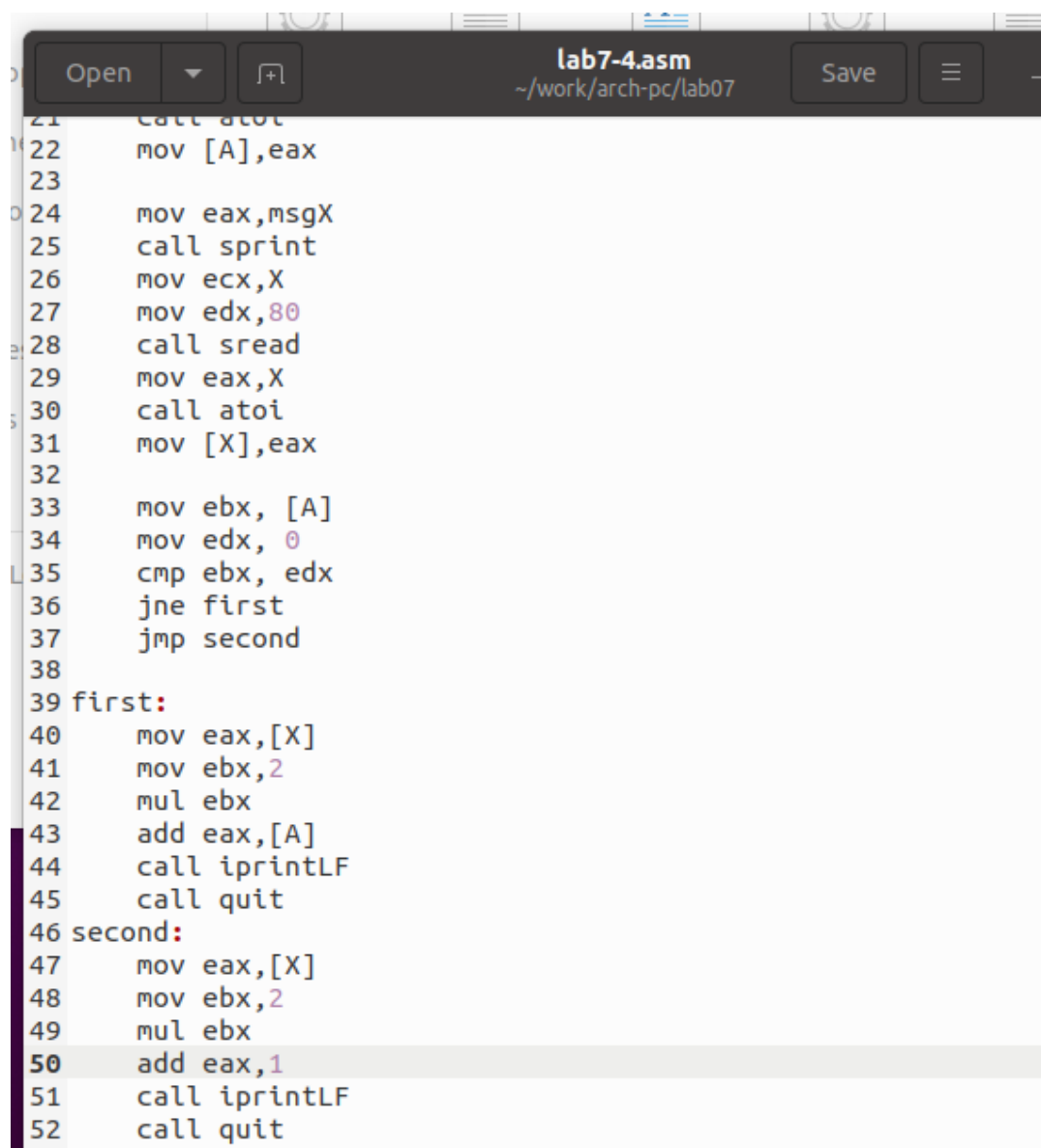
Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений X и a из 7.6. (рис. 2.14) (рис. 2.15)

для варианта 4

$$\begin{cases} 2x + a, a \neq 0 \\ 2x + 1, a = 0 \end{cases}$$

При $x = 3, a = 0$ получается 7.

При $x = 3, a = 2$ получается 8.



```
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, [A]
34      mov edx, 0
35      cmp ebx, edx
36      jne first
37      jmp second
38
39 first:
40      mov eax,[X]
41      mov ebx,2
42      mul ebx
43      add eax,[A]
44      call iprintLF
45      call quit
46 second:
47      mov eax,[X]
48      mov ebx,2
49      mul ebx
50      add eax,1
51      call iprintLF
52      call quit
```

Рис. 2.14: Программа lab7-4.asm

```
timursindyanov@Ubuntu:~/work/arch-pc/lab07$  
timursindyanov@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm  
timursindyanov@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4  
timursindyanov@Ubuntu:~/work/arch-pc/lab07$ ./lab7-4  
Input A: 0  
Input X: 3  
7  
timursindyanov@Ubuntu:~/work/arch-pc/lab07$ ./lab7-4  
Input A: 2  
Input X: 3  
8  
timursindyanov@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.15: Запуск программы lab7-4.asm

3 Выводы

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