

# **Отчёт по лабораторной работе 7**

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

Лянь Цзэюй

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	20

## 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 . . . . .	12
2.9	Файл листинга lab7-2 . . . . .	13
2.10	Ошибка трансляции lab7-2 . . . . .	14
2.11	Файл листинга с ошибкой lab7-2 . . . . .	15
2.12	Программа в файле task.asm . . . . .	16
2.13	Запуск программы task.asm . . . . .	16
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 программам 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.

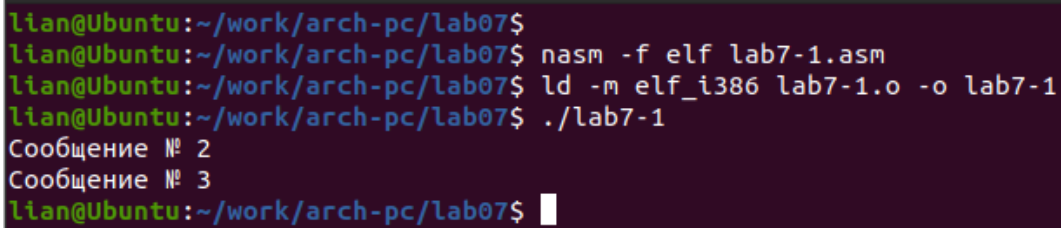
```

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 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 в файле lab7-1.asm

Создал исполняемый файл и запустил его.

A terminal window with a dark background and light-colored text. The prompt is 'lian@Ubuntu:~/work/arch-pc/lab07\$'. The user enters 'nasm -f elf lab7-1.asm', followed by 'ld -m elf\_i386 lab7-1.o -o lab7-1', and then './lab7-1'. The output shows 'Сообщение № 2', 'Сообщение № 3', and the prompt again.

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



```

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 sprintf
15 jmp _end
16
17 _label2:
18 mov eax, msg2
19 call sprintf
20 jmp _label1
21
22 _label3:
23 mov eax, msg3
24 call sprintf
25
26 _end:
27 call quit

```

Рис. 2.3: Программа в файле lab7-1.asm:

```

lian@Ubuntu:~/work/arch-pc/lab07$
lian@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
lian@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
lian@Ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
lian@Ubuntu:~/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 sprintf
15 jmp _end
16
17 _label2:
18 mov eax, msg2
19 call sprintf
20 jmp _label1
21
22 _label3:
23 mov eax, msg3
24 call sprintf
25 jmp _label2
26
27 _end:
28 call quit
```

Рис. 2.5: Программа в файле lab7-1.asm

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

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

```

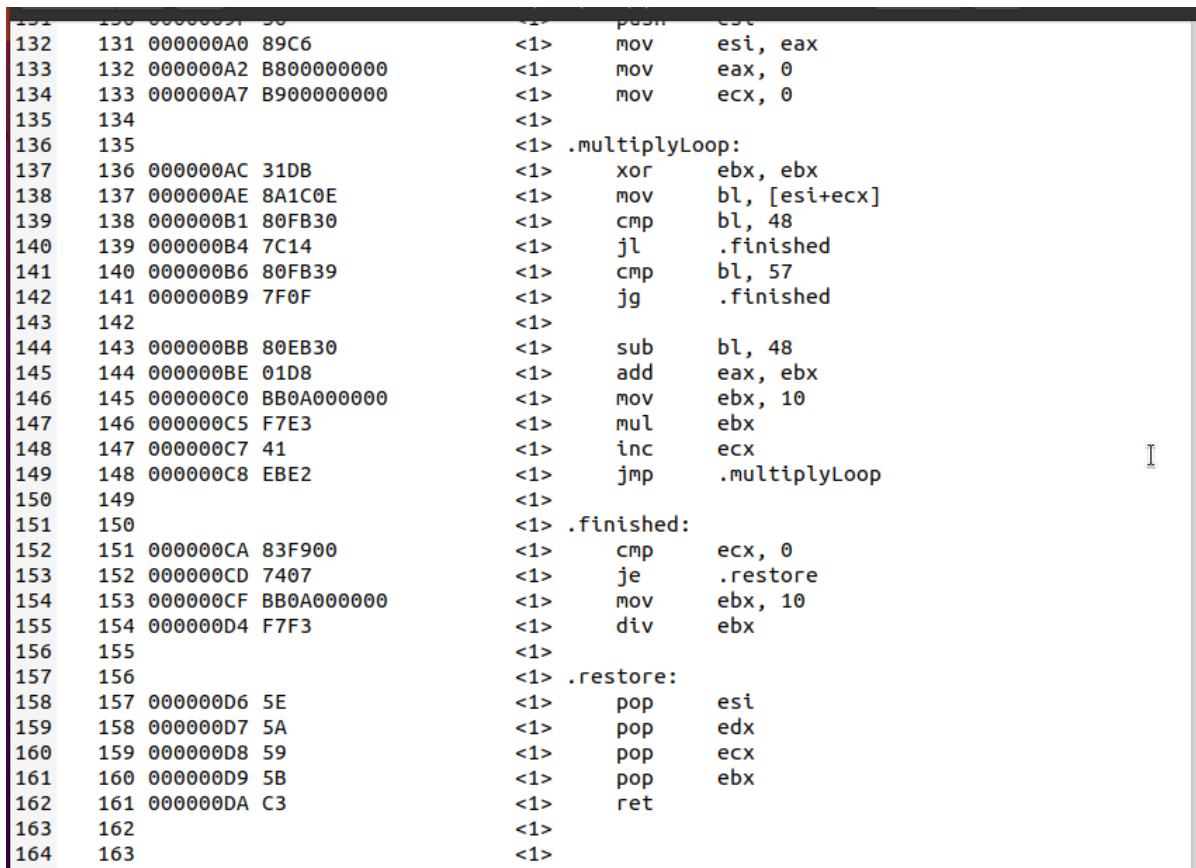
lian@Ubuntu:~/work/arch-pc/lab07$
lian@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
lian@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
lian@Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
lian@Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 20
Наибольшее число: 50
lian@Ubuntu:~/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



```
132 131 000000A0 89C6 <1> push esi, eax
133 132 000000A2 B800000000 <1> mov eax, 0
134 133 000000A7 B900000000 <1> mov ecx, 0
135 134 <1>
136 135 <1> .multiplyLoop:
137 136 000000AC 31DB <1> xor ebx, ebx
138 137 000000AE 8A1C0E <1> mov bl, [esi+ecx]
139 138 000000B1 80FB30 <1> cmp bl, 48
140 139 000000B4 7C14 <1> jl .finished
141 140 000000B6 80FB39 <1> cmp bl, 57
142 141 000000B9 7F0F <1> jg .finished
143 142 <1>
144 143 000000BB 80EB30 <1> sub bl, 48
145 144 000000BE 01D8 <1> add eax, ebx
146 145 000000C0 BB0A000000 <1> mov ebx, 10
147 146 000000C5 F7E3 <1> mul ebx
148 147 000000C7 41 <1> inc ecx
149 148 000000C8 EBE2 <1> jmp .multiplyLoop
150 149 <1>
151 150 <1> .finished:
152 151 000000CA 83F900 <1> cmp ecx, 0
153 152 000000CD 7407 <1> je .restore
154 153 000000CF BB0A000000 <1> mov ebx, 10
155 154 000000D4 F7F3 <1> div ebx
156 155 <1>
157 156 <1> .restore:
158 157 000000D6 5E <1> pop esi
159 158 000000D7 5A <1> pop edx
160 159 000000D8 59 <1> pop ecx
161 160 000000D9 5B <1> pop ebx
162 161 000000DA C3 <1> ret
163 162 <1>
164 163 <1>
```

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

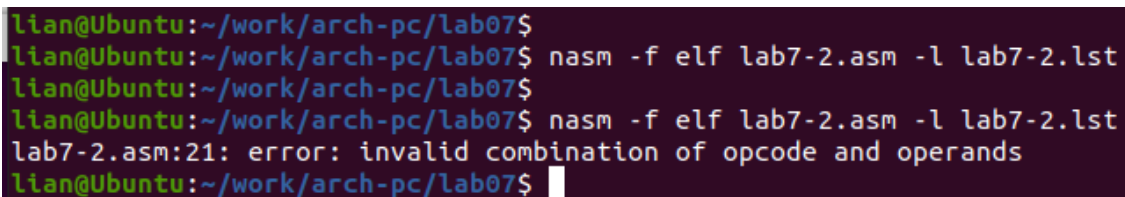
строка 212

- 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.

A screenshot of a terminal window with a dark background and light-colored text. The prompt is 'lian@Ubuntu:~/work/arch-pc/lab07\$'. The user enters 'nasm -f elf lab7-2.asm -l lab7-2.lst' twice. The first time, it runs successfully. The second time, it shows an error: 'lab7-2.asm:21: error: invalid combination of opcode and operands'.

```
lian@Ubuntu:~/work/arch-pc/lab07$  
lian@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
lian@Ubuntu:~/work/arch-pc/lab07$  
lian@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
lab7-2.asm:21: error: invalid combination of opcode and operands  
lian@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.10: Ошибка трансляции lab7-2

```

190 15 000000ED E81DFFFFFF call sprintf
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 mov eax,
197 21 ***** error: invalid combination of opcode and
operands
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

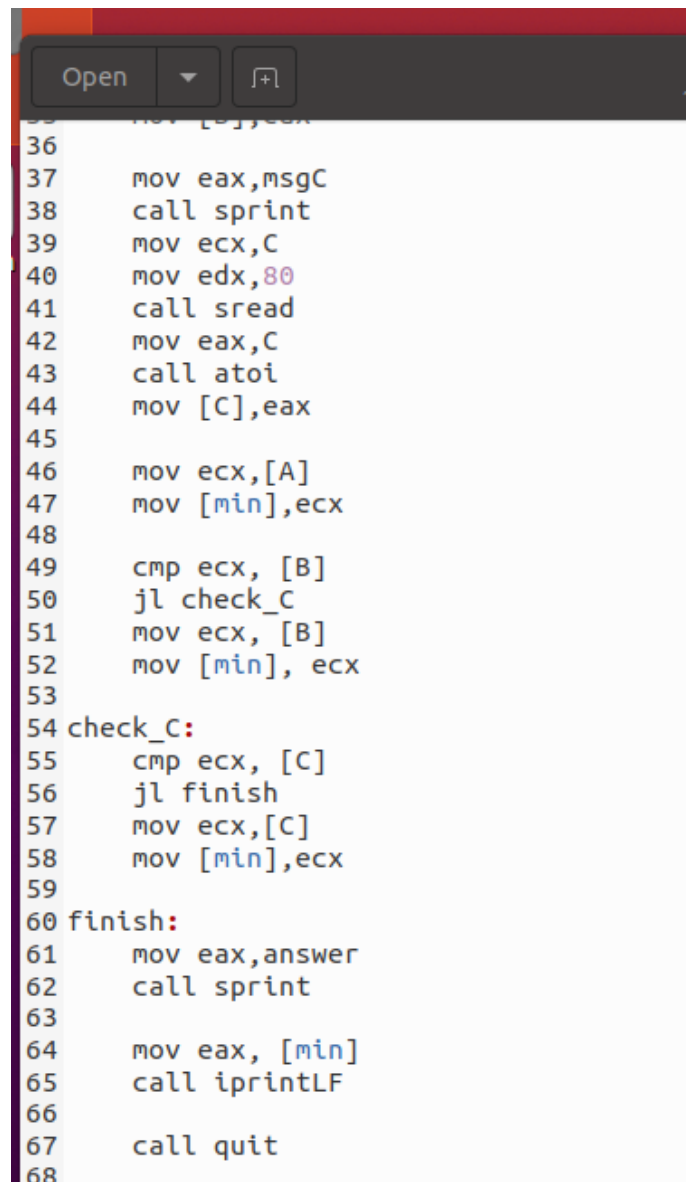
```

Рис. 2.11: Файл листинга с ошибкой lab7-2

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

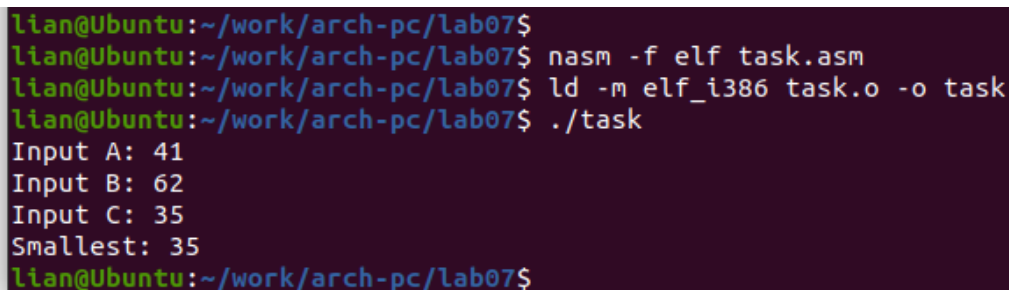
5. Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу

для варианта 10 - 41,62,35



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

Рис. 2.12: Программа в файле task.asm



```
lian@Ubuntu:~/work/arch-pc/lab07$
lian@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf task.asm
lian@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 task.o -o task
lian@Ubuntu:~/work/arch-pc/lab07$ ./task
Input A: 41
Input B: 62
Input C: 35
Smallest: 35
lian@Ubuntu:~/work/arch-pc/lab07$
```

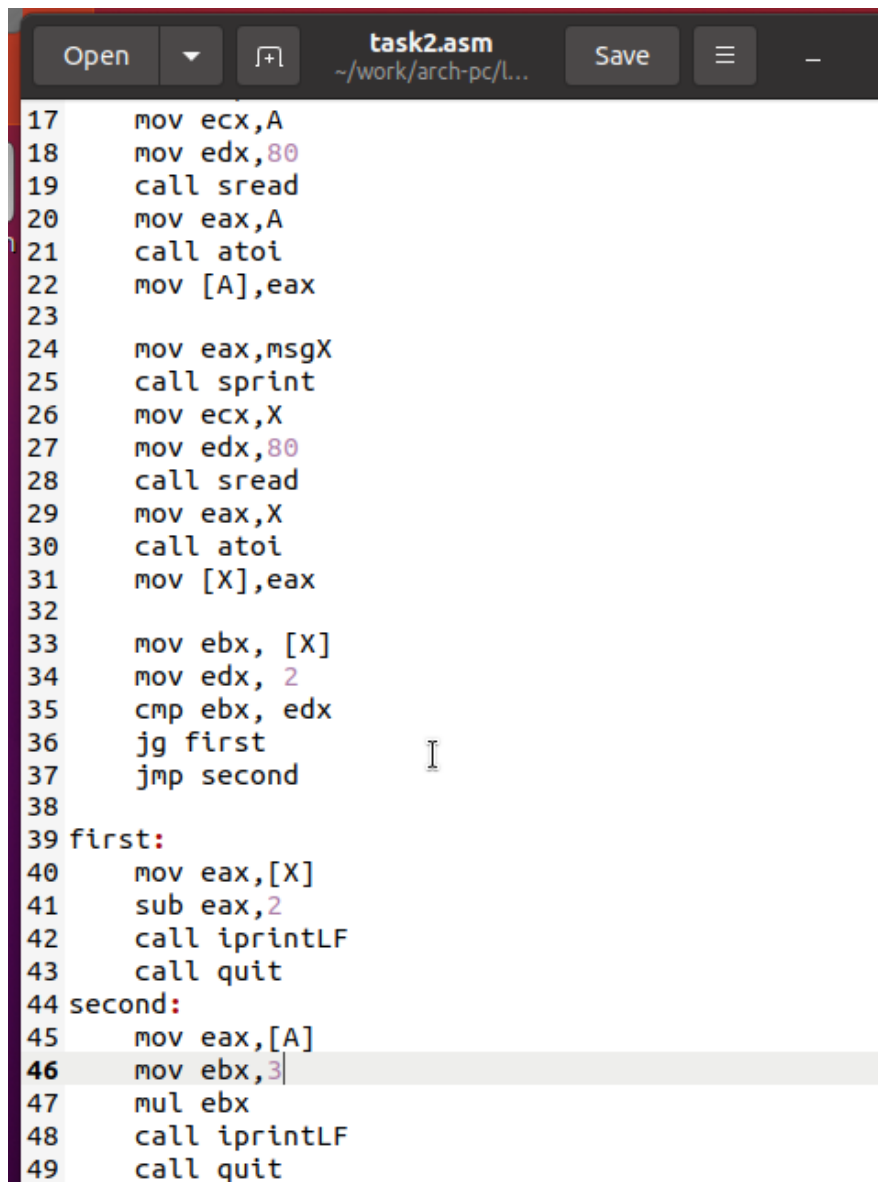
Рис. 2.13: Запуск программы task.asm



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 10

$$\begin{cases} x - 2, x > 2 \\ 3a, x \leq 2 \end{cases}$$



```
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 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, 2
35     cmp ebx, edx
36     jg first
37     jmp second
38
39 first:
40     mov eax,[X]
41     sub eax,2
42     call iprintLF
43     call quit
44 second:
45     mov eax,[A]
46     mov ebx,3
47     mul ebx
48     call iprintLF
49     call quit
```

Рис. 2.14: Программа в файле task2.asm

```
lian@Ubuntu:~/work/arch-pc/lab07$  
lian@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf task2.asm  
lian@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 task2.o -o task2  
lian@Ubuntu:~/work/arch-pc/lab07$ ./task2  
Input A: 0  
Input X: 3  
1  
lian@Ubuntu:~/work/arch-pc/lab07$ ./task2  
Input A: 2  
Input X: 1  
6  
lian@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.15: Запуск программы task2.asm

## 3 Выводы

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