

# **Отчёт по лабораторной работе 7**

**дисциплина: Архитектура компьютера**

Шангина В. А НКАбд-05-24

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Реализация переходов в NASM . . . . .	6
2.2	Условные переходы . . . . .	11
2.3	Изучение структуры файла листинга . . . . .	13
2.4	Самостоятельное задание . . . . .	16
<b>3</b>	<b>Выводы</b>	<b>21</b>

## Список иллюстраций

2.1	Создан каталог . . . . .	6
2.2	Программа lab7-1.asm . . . . .	7
2.3	Запуск программы lab7-1.asm . . . . .	8
2.4	Программа lab7-1.asm . . . . .	9
2.5	Запуск программы lab7-1.asm . . . . .	9
2.6	Программа lab7-1.asm . . . . .	10
2.7	Запуск программы lab7-1.asm . . . . .	11
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 task1.asm . . . . .	17
2.14	Запуск программы task1.asm . . . . .	17
2.15	Программа task2.asm . . . . .	19
2.16	Запуск программы task2.asm . . . . .	20

## **Список таблиц**

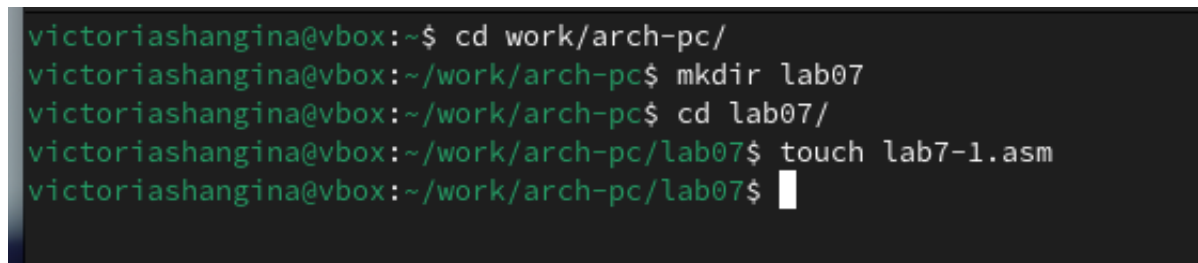
# 1 Цель работы

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)



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

Рис. 2.1: Создан каталог

Инструкция `jmp` в NASM используется для реализации безусловных переходов. Пример программы, демонстрирующей эту инструкцию, приведен в файле lab7-1.asm. (рис. 2.2)

```

%include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
jmp _label2

_label1:
mov eax, msg1
call printf

_label2:
mov eax, msg2
call printf

_label3:
mov eax, msg3
call printf

_end:
call quit

```

Рис. 2.2: Программа lab7-1.asm

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

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

Рис. 2.3: Запуск программы lab7-1.asm

Инструкция `jmp` позволяет осуществлять переходы как вперед, так и назад. Для изменения последовательности вывода программы добавляю метки `_label1` и `_end`. Таким образом, вывод программы изменится: сначала отобразится сообщение № 2, затем сообщение № 1, и программа завершит работу.

Обновляю текст программы согласно листингу 7.2. (рис. 2.4, рис. 2.5)



```

%include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
jmp _label2

_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF

_end:
call quit

```

Рис. 2.4: Программа lab7-1.asm

```

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

```

Рис. 2.5: Запуск программы lab7-1.asm

Дорабатываю текст программы для вывода следующих сообщений:

Сообщение № 3

Сообщение № 2

Сообщение № 1

Результат показан на рисунках (рис. 2.6, рис. 2.7).

```
%include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
jmp _label3

_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit
```

Рис. 2.6: Программа lab7-1.asm

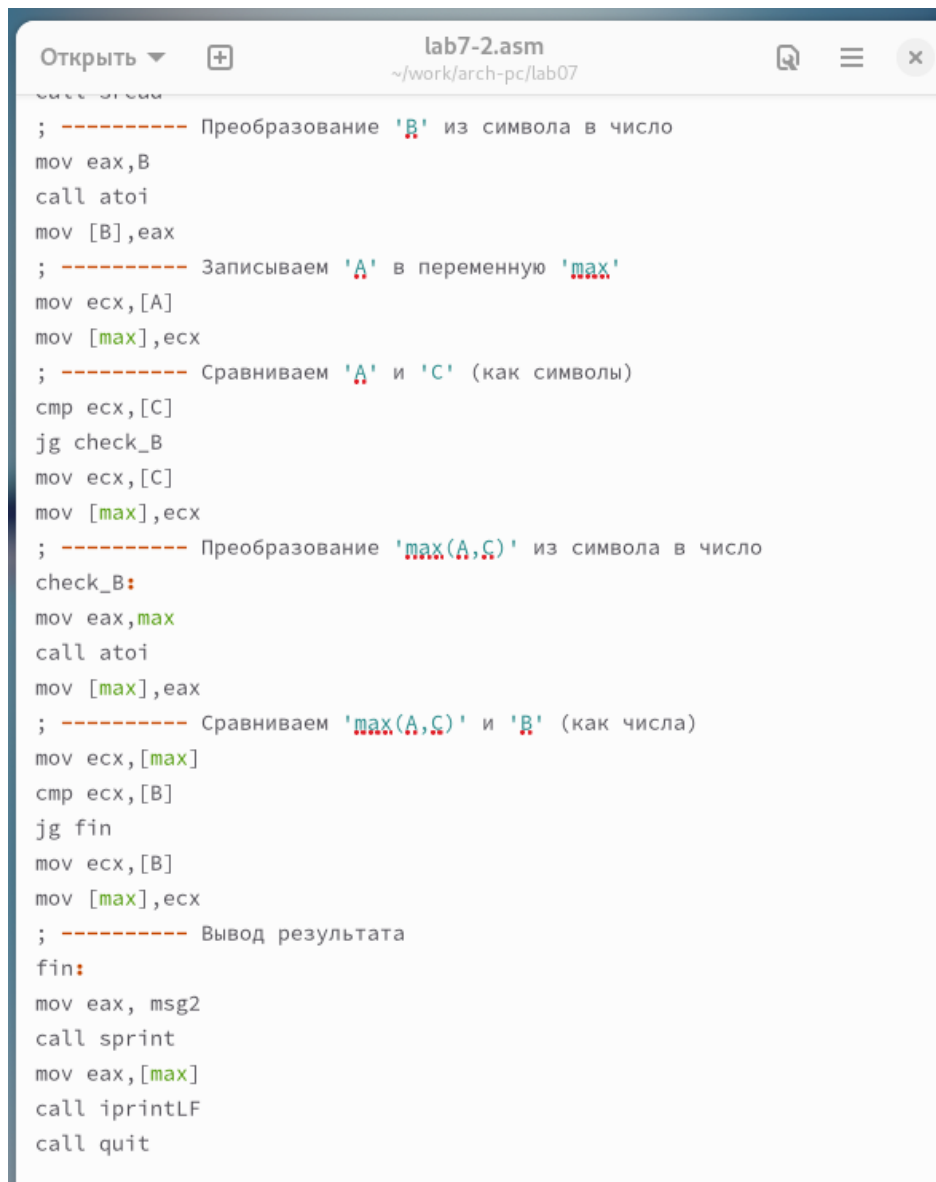
```
victoriashangina@vbox:~/work/arch-pc/lab07$  
victoriashangina@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm  
victoriashangina@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1  
victoriashangina@vbox:~/work/arch-pc/lab07$ ./lab7-1  
Сообщение № 3  
Сообщение № 2  
Сообщение № 1 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

Рис. 2.7: Запуск программы lab7-1.asm

Использование инструкции `jmp` обеспечивает переходы независимо от условий. Однако для реализации условных переходов требуется использование дополнительных инструкций.

## 2.2 Условные переходы

Для демонстрации условных переходов создаю программу, определяющую максимальное значение среди трех переменных: А, В и С. Значения А и С задаются в программе, а В вводится с клавиатуры. Результаты работы программы представлены на рисунках (рис. 2.8, рис. 2.9).



```
Открыть + lab7-2.asm ~\work\arch-pc\lab07
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A]
mov [max],ecx
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi
mov [max],eax
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
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. 2.8: Программа lab7-2.asm

```
victoriashangina@vbox:~/work/arch-pc/lab07$  
victoriashangina@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm  
victoriashangina@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2  
victoriashangina@vbox:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 10  
Наибольшее число: 50  
victoriashangina@vbox:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 20  
Наибольшее число: 50  
victoriashangina@vbox:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 60  
Наибольшее число: 60  
victoriashangina@vbox:~/work/arch-pc/lab07$
```

Рис. 2.9: Запуск программы lab7-2.asm

## 2.3 Изучение структуры файла листинга

Для получения файла листинга указываю ключ `-l` при ассемблировании. Результат ассемблирования программы lab7-2.asm представлен на рисунке (рис. 2.10).

```

188 13 ; ----- Вывод сообщения 'Введите B: '
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 F801FFFFFF 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 ig 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 F862FFFFFF 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 ig fin
216 41 0000014D 8B0D[0A000000] mov ecx,[B]
217 42 00000153 890D[00000000] mov [max],ecx
218 43 ; ----- Вывод результата
219 44 fin:

```

Рис. 2.10: Файл листинга lab7-2

Анализируя структуру листинга, можно увидеть соответствие строк кода и их машинного представления. Например:

- **Строка 203:**
  - Номер строки: 28
  - Адрес: 0000011C
  - Машинный код: 3B0D[39000000]
  - Команда: `cmp ecx,[C]`

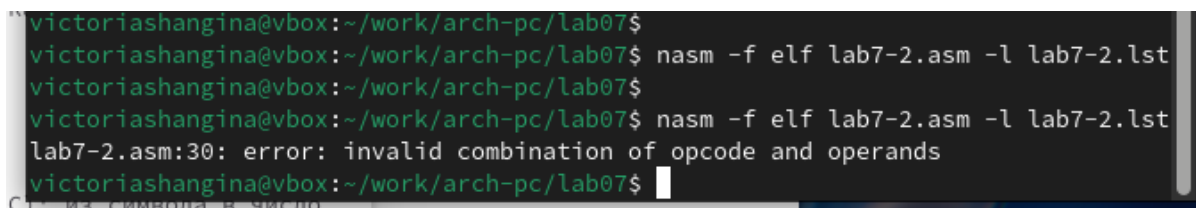
- **Строка 204:**

- Номер строки: 29
- Адрес: 00000122
- Машинный код: 7F0C
- Команда: jg check\_V

- **Строка 205:**

- Номер строки: 30
- Адрес: 00000124
- Машинный код: 8B0D[39000000]
- Команда: mov ecx,[C]

Далее изменяю инструкцию с двумя операндами, удаляя один, и повторяю трансляцию. Возникает ошибка, результат которой отображен на рисунках (рис. 2.11, рис. 2.12).

A screenshot of a terminal window with a dark background and green text. The terminal shows the user 'victoriashangina' at a 'vbox' machine in the directory '~/work/arch-pc/lab07'. They run the command 'nasm -f elf lab7-2.asm -l lab7-2.lst' twice. The first run is successful. The second run produces an error: 'lab7-2.asm:30: error: invalid combination of opcode and operands'. The prompt is ready for the next command.

```
victoriashangina@vbox:~/work/arch-pc/lab07$  
victoriashangina@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
victoriashangina@vbox:~/work/arch-pc/lab07$  
victoriashangina@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
lab7-2.asm:30: error: invalid combination of opcode and operands  
victoriashangina@vbox:~/work/arch-pc/lab07$
```

Рис. 2.11: Ошибка трансляции lab7-2

```

196 21 00000101 B8[0A000000] mov eax,B
197 22 00000106 F801FFFFFF 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 7F06 ig check_B
205 30 mov ecx,
206 30 ***** error: invalid combination of opcode and operands
207 31 00000124 890D[00000000] mov [max],ecx
208 32 ; ----- Преобразование 'max(A,C)' из символа в число
209 33 check_B:
210 34 0000012A B8[00000000] mov eax,max
211 35 0000012F F868FFFFFF call atoi
212 36 00000134 A3[00000000] mov [max],eax
213 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
214 38 00000139 8B0D[00000000] mov ecx,[max]
215 39 0000013F 3B0D[0A000000] cmp ecx,[B]
216 40 00000145 7F0C ig fin
217 41 00000147 8B0D[0A000000] mov ecx,[B]
218 42 0000014D 890D[00000000] mov [max],ecx
219 43 ; ----- Вывод результата
220 44 fin:
221 45 00000153 B8[13000000] mov eax,msg2
222 46 00000158 F8B2FFFFFF call sprintf
223 47 0000015D A1[00000000] mov eax,[max]
224 48 00000162 F81FFFFFFF call iprintf
225 49 00000167 F86FFFFFFF call quit

```

Рис. 2.12: Файл листинга с ошибкой lab7-2

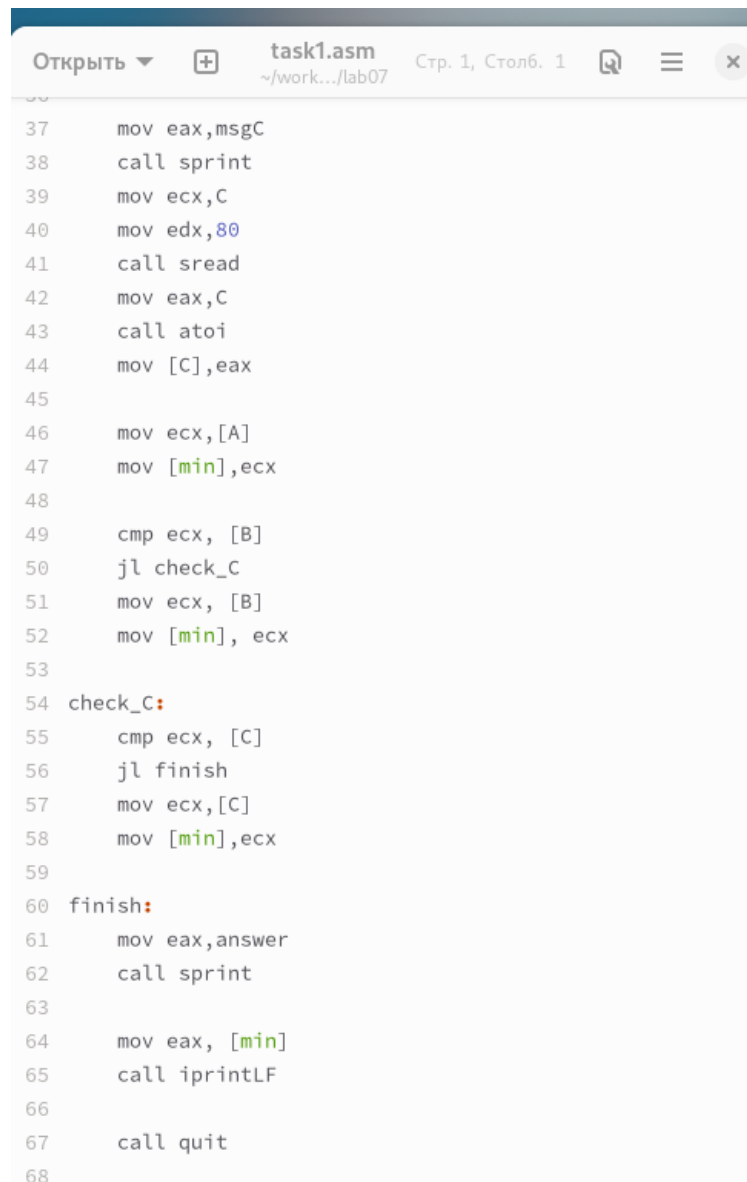
## 2.4 Самостоятельное задание

1. Напишите программу, которая находит наименьшее значение из трех переменных a, b и c для следующих значений:

**Вариант 11:** 21,28,34.

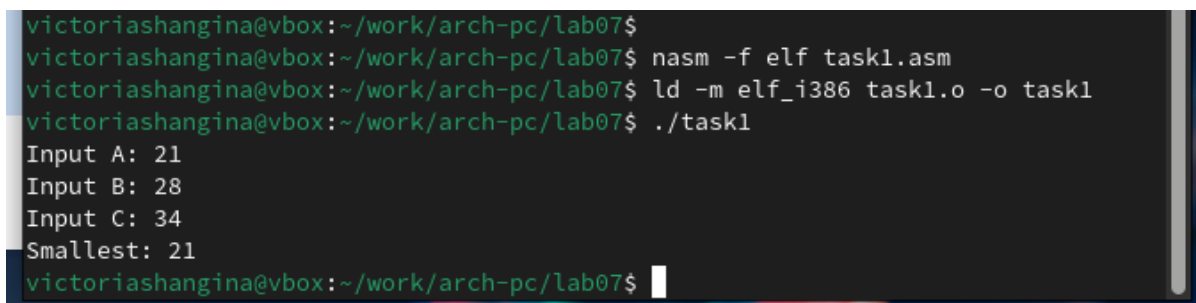
Результат работы программы показан на рисунках (рис. 2.13, рис. 2.14).





```
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. 2.13: Программа task1.asm



```
victoriashangina@vbox:~/work/arch-pc/lab07$
victoriashangina@vbox:~/work/arch-pc/lab07$ nasm -f elf task1.asm
victoriashangina@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 task1.o -o task1
victoriashangina@vbox:~/work/arch-pc/lab07$ ./task1
Input A: 21
Input B: 28
Input C: 34
Smallest: 21
victoriashangina@vbox:~/work/arch-pc/lab07$
```

Рис. 2.14: Запуск программы task1.asm

2. Напишите программу для вычисления функции  $f(x)$  для введенных значений  $x$  и  $a$ :

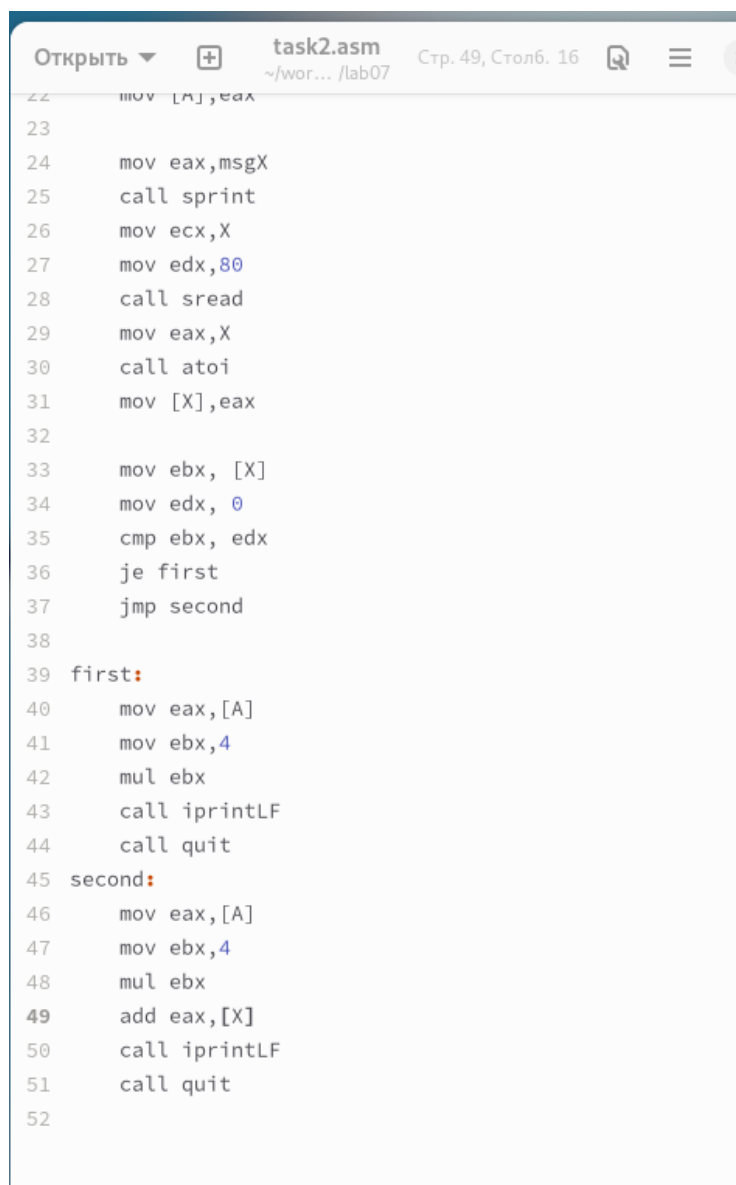
**Вариант 11:**

$$f(x) = \begin{cases} 4a, & \text{если } x = 0 \\ 4a + x, & \text{если } x \neq 0 \end{cases}$$

При  $x = 0, a = 3$  результат: 12.

При  $x = 1, a = 2$  результат: 9.

Результаты программы представлены на рисунках (рис. 2.15, рис. 2.16).



```
task2.asm
~/wor... /lab07
Стр. 49, Столб. 16

22     mov [X],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, 0
35     cmp ebx, edx
36     je first
37     jmp second
38
39 first:
40     mov eax,[A]
41     mov ebx,4
42     mul ebx
43     call iprintLF
44     call quit
45 second:
46     mov eax,[A]
47     mov ebx,4
48     mul ebx
49     add eax,[X]
50     call iprintLF
51     call quit
52
```

Рис. 2.15: Программа task2.asm

```
victoriashangina@vbox:~/work/arch-pc/lab07$  
victoriashangina@vbox:~/work/arch-pc/lab07$ nasm -f elf task2.asm  
victoriashangina@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 task2.o -o task2  
victoriashangina@vbox:~/work/arch-pc/lab07$ ./task2  
Input A:  
Input X:  
0  
victoriashangina@vbox:~/work/arch-pc/lab07$  
victoriashangina@vbox:~/work/arch-pc/lab07$ ./task2  
Input A: 3  
Input X: 0  
12  
victoriashangina@vbox:~/work/arch-pc/lab07$ ./task2  
Input A: 2  
Input X: 1  
9  
victoriashangina@vbox:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы task2.asm

## 3 Выводы

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