

# **Отчёт по лабораторной работе 7**

**Архитектура компьютеров**

Рамиэль Сарханов

# 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	Изучение структуры файла листинга . . . . .	13
2.3	Самостоятельное задание . . . . .	15
<b>3</b>	<b>Выводы</b>	<b>20</b>

## Список иллюстраций

2.1	Создан каталог . . . . .	6
2.2	Программа lab7-1.asm . . . . .	7
2.3	Запуск программы lab7-1.asm . . . . .	7
2.4	Программа lab7-1.asm . . . . .	8
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 . . . . .	12
2.10	Файл листинга lab7-2 . . . . .	13
2.11	Ошибка трансляции lab7-2 . . . . .	14
2.12	Файл листинга с ошибкой lab7-2 . . . . .	15
2.13	Программа lab7-task1.asm . . . . .	16
2.14	Запуск программы lab7-task1.asm . . . . .	16
2.15	Программа lab7-task2.asm . . . . .	18
2.16	Запуск программы lab7-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

### 2.1 Реализация переходов в NASM

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



Рис. 2.1: Создан каталог

В NASM инструкция `jmp` используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. В файле lab7-1.asm разместил 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 printf
15
16 _label2:
17 mov eax, msg2
18 call printf
19
20 _label3:
21 mov eax, msg3
22 call printf
23
24 _end:
25 call quit

```

Рис. 2.2: Программа lab7-1.asm

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

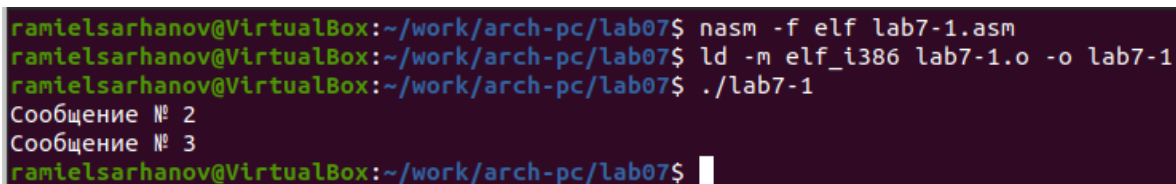
```

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

```

Рис. 2.3: Запуск программы lab7-1.asm

Инструкция `jmp` позволяет выполнять переходы как вперёд, так и назад. Изменил программу так, чтобы сначала выводилось сообщение № 2, затем сообщение № 1, после чего программа завершала работу. Для этого добавил в текст программы инструкцию `jmp` с меткой `_label1` после вывода сообщения № 2 (чтобы перейти к инструкции вывода сообщения № 1) и инструкцию `jmp` с меткой `_end` после вывода сообщения № 1 (для перехода к инструкции `call quit`). Обновил текст программы согласно листингу 7.2 (рис. 2.4 и 2.5).



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

Рис. 2.4: 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. 2.5: Запуск программы lab7-1.asm

Изменил текст программы так, чтобы итоговый вывод программы выглядел следующим образом (рис. 2.6 и 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 lab7-1.asm

```

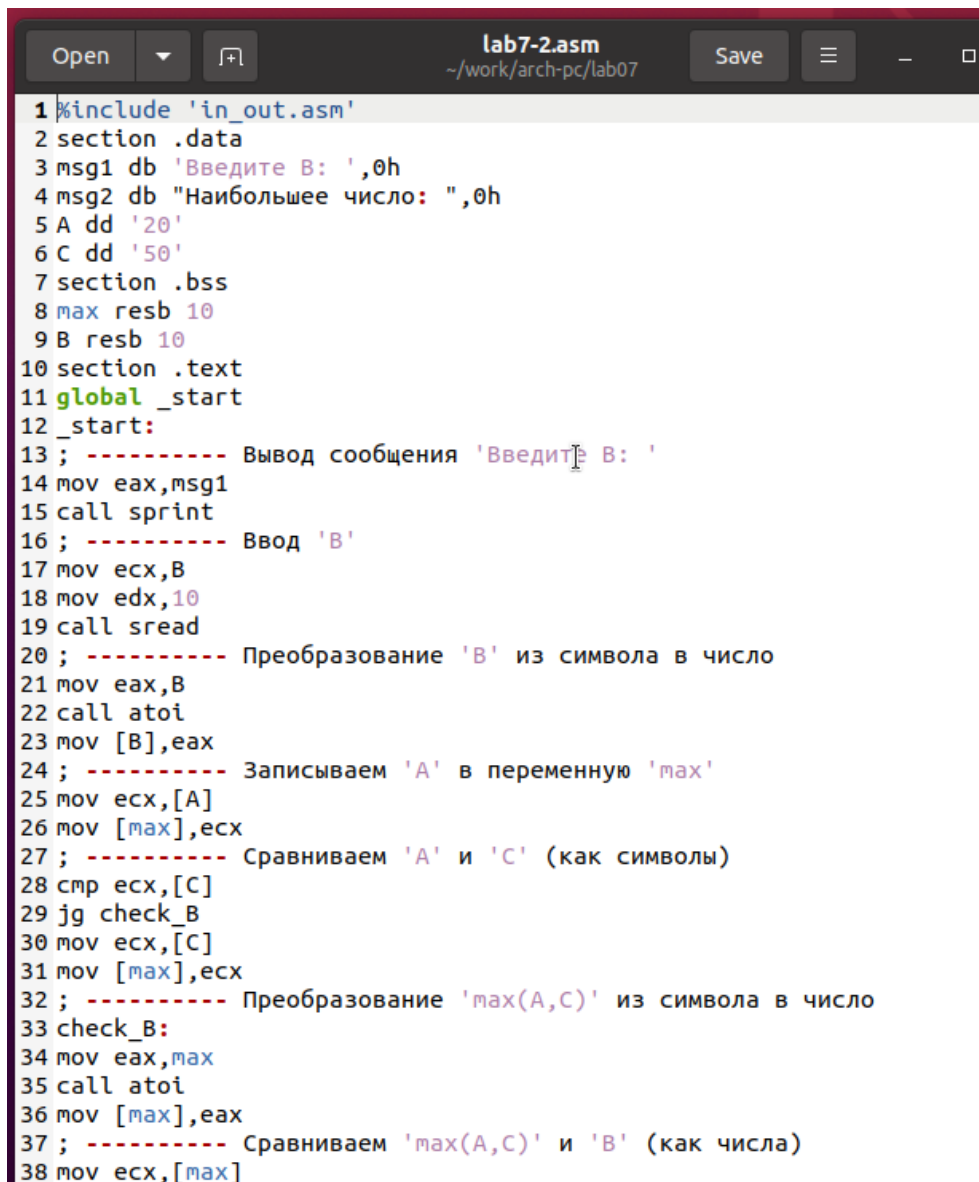
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$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

```

Рис. 2.7: Запуск программы lab7-1.asm

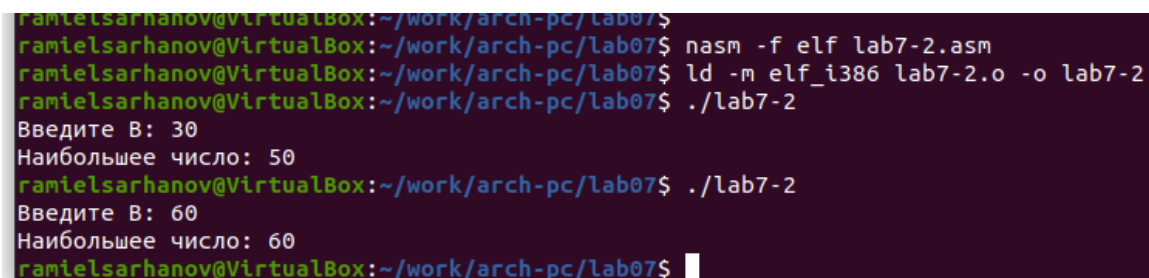
Инструкция `jmp` всегда вызывает переход. Однако часто в программировании требуются условные переходы, которые выполняются только при соблюдении определённых условий. В качестве примера рассмотрим программу, определяющую и выводящую наибольшее значение среди трёх целочисленных переменных `A`, `B` и `C`. Значения для `A` и `C` заданы в программе, а `B` вводится с клавиатуры.

Создал исполняемый файл и проверил его работу для различных значений `B` (рис. 2.8 и 2.9).



```
1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '20'
6 C dd '50'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax,msg1
15 call sprint
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]
```

Рис. 2.8: Программа lab7-2.asm



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

Рис. 2.9: Запуск программы lab7-2.asm

## 2.2 Изучение структуры файла листинга

Обычно NASM создаёт только объектный файл. Чтобы получить файл листинга, нужно указать ключ `-l` и задать имя файла листинга в командной строке.

Создал файл листинга для программы из файла `lab7-2.asm` (рис. 2.10).

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
221	46	0000015E	E8ACFFFFFF	call sprint
222	47	00000163	A1[00000000]	mov eax,[max]
223	48	00000168	E819FFFFFF	call iprintLF
224	49	0000016D	E869FFFFFF	call quit

Рис. 2.10: Файл листинга lab7-2

Рассмотрим его структуру:

- **Строка 211**

- 36 — номер строки
- 0000013A — адрес
- A3[00000000] — машинный код
- `mov [max],eax` — код программы

- **Строка 213**

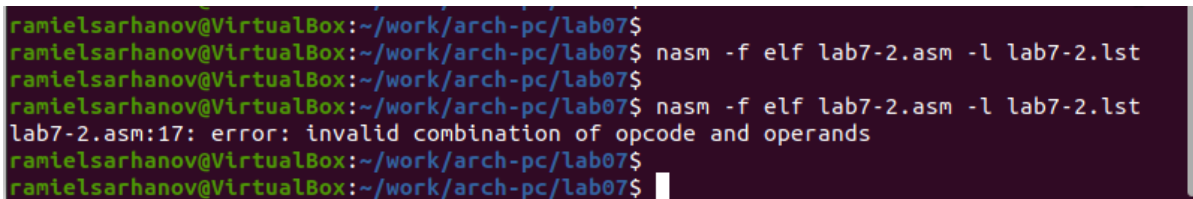
- 38 — номер строки
- 0000013F — адрес
- 8B0D[00000000] — машинный код

- `mov esx,[max]` — код программы

- **Строка 219**

- 39 — номер строки
- 00000145 — адрес
- 3B0D[0A000000] — машинный код
- `cmp esx,[B]` — код программы

Открыл файл `lab7-2.asm`, удалил один из операндов в инструкции с двумя операндами и выполнил трансляцию с получением файла листинга (рис. 2.11 и 2.12).



```
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$  
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$  
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
lab7-2.asm:17: error: invalid combination of opcode and operands  
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$  
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.11: Ошибка трансляции `lab7-2`

```

190 15 000000ED E81DFFFFFF      call sprint
191 16                          ; ----- Ввод 'B'
192 17                          mov ecx,
193 17 *****
194 18 000000F2 BA0A000000      error: invalid combination of opcode and operands
195 19 000000F7 E847FFFFFF      mov edx,10
196 20                          call sread
197 21 000000FC B8[0A000000]    ; ----- Преобразование 'B' из символа в число
198 22 00000101 E896FFFFFF      mov eax,B
199 23 00000106 A3[0A000000]    call atoi
200 24                          mov [B],eax
201 25 0000010B 8B0D[35000000]    ; ----- Записываем 'A' в переменную 'max'
202 26 00000111 890D[00000000]    mov ecx,[A]
203 27                          mov [max],ecx
204 28 00000117 3B0D[39000000]    ; ----- Сравниваем 'A' и 'C' (как символы)
205 29 0000011D 7F0C            cmp ecx,[C]
206 30 0000011F 8B0D[39000000]    jg check_B
207 31 00000125 890D[00000000]    mov ecx,[C]
208 32                          mov [max],ecx
209 33                          ; ----- Преобразование 'max(A,C)' из символа в число
210 34 0000012B B8[00000000]    check_B:
211 35 00000130 E867FFFFFF      mov eax,max
212 36 00000135 A3[00000000]    call atoi
213 37                          mov [max],eax
214 38 0000013A 8B0D[00000000]    ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
215 39 00000140 3B0D[0A000000]    mov ecx,[max]
216 40 00000146 7F0C            cmp ecx,[B]
217 41 00000148 8B0D[0A000000]    jg fin
218 42 0000014E 890D[00000000]    mov ecx,[B]
219 43                          mov [max],ecx
220 44                          ; ----- Вывод результата
221 45 00000154 B8[13000000]    fin:
222 46 00000159 E8B1FEFFFF      mov eax,msg2
223 47 0000015E A1[00000000]    call sprint
224 48 00000163 E81EFFFFFF      mov eax,[max]
225 49 00000168 8B0D[00000000]    call iprintf
226 50 0000016E 8B0D[00000000]    call iprintf

```

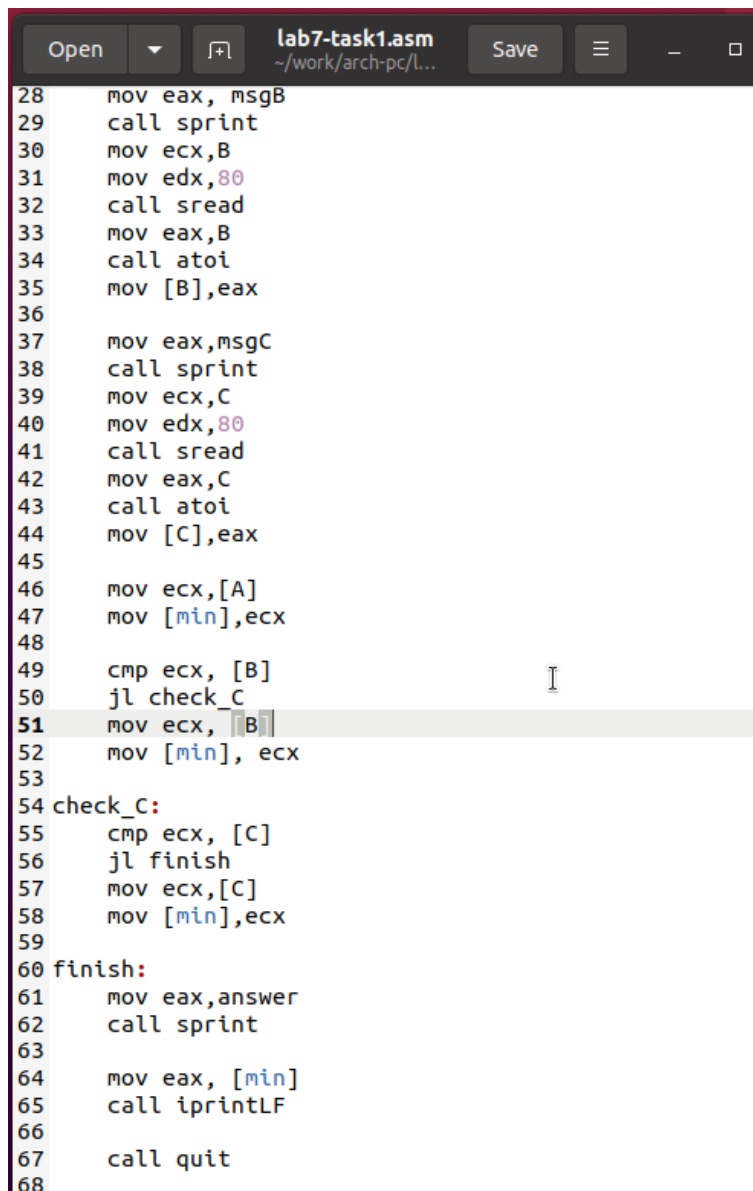
Рис. 2.12: Файл листинга с ошибкой lab7-2

Из-за ошибки объектный файл не был создан. Однако, листинг указал местоположение ошибки.

## 2.3 Самостоятельное задание

1. Найти наименьшее среди трёх целочисленных переменных  $a$ ,  $b$  и  $c$ , используя значения из таблицы 7.5 для варианта, полученного при выполнении лабораторной работы № 6. Создать исполняемый файл и проверить его работу (рис. 2.13 и 2.14).

Для варианта 7:  $a = 45$ ,  $b = 67$ ,  $c = 15$ .

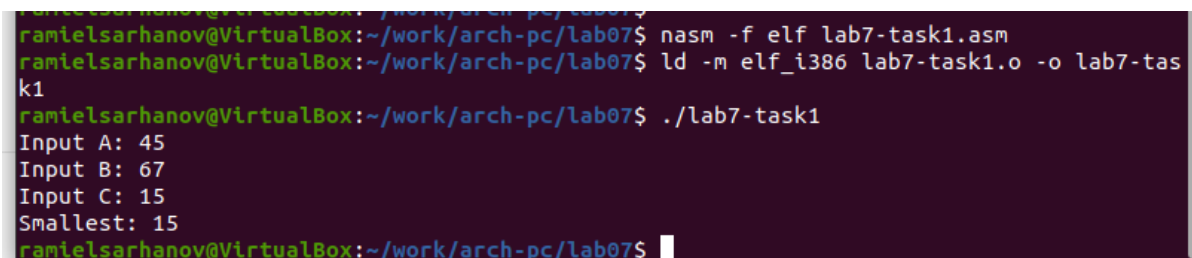


```

28  mov eax, msgB
29  call sprint
30  mov ecx, B
31  mov edx, 80
32  call sread
33  mov eax, B
34  call atoi
35  mov [B], eax
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. 2.13: Программа lab7-task1.asm



```

ramielsarhanov@VirtualBox: ~/work/arch-pc/lab07$ nasm -f elf lab7-task1.asm
ramielsarhanov@VirtualBox: ~/work/arch-pc/lab07$ ld -m elf_i386 lab7-task1.o -o lab7-task1
ramielsarhanov@VirtualBox: ~/work/arch-pc/lab07$ ./lab7-task1
Input A: 45
Input B: 67
Input C: 15
Smallest: 15
ramielsarhanov@VirtualBox: ~/work/arch-pc/lab07$

```

Рис. 2.14: Запуск программы lab7-task1.asm



2. **Программа для вычисления функции  $f(x)$**  при введенных значениях  $x$  и  $a$  с клавиатуры. Вид функции  $f(x)$  выбирается из таблицы 7.6 в зависимости от варианта, полученного для лабораторной работы № 7. Создать исполняемый файл и проверить его работу для значений  $x$  и  $a$  из таблицы 7.6 (рис. 2.15 и 2.16).

Для варианта 7:

$$f(x) = \begin{cases} 6a, & x = a \\ a + x, & x \neq a \end{cases}$$

При  $x = 1, a = 1$  результат — 6.

При  $x = 2, a = 1$  результат — 3.

```

9      result:      RESB 80
10
11 SECTION .text
12     GLOBAL _start
13
14 _start:
15     mov eax,msgA
16     call sprint
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, [A]
35     cmp ebx, edx
36     je first
37     jmp second
38
39 first:
40     mov eax,[A]
41     mov ebx,6
42     mul ebx
43     call iprintLF
44     call quit
45 second:
46     mov eax,[X]
47     add eax,[A]
48     call iprintLF
49     call quit
50

```

Рис. 2.15: Программа lab7-task2.asm

```
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$  
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-task2.asm  
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-task2.o -o lab7-task2  
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$ ./lab7-task2  
Input A: 1  
Input X: 1  
6  
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$ ./lab7-task2  
Input A: 1  
Input X: 2  
3  
ramielsarhanov@VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы lab7-task2.asm

## 3 Выводы

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