

# **Отчёт по лабораторной работе 7**

**Архитектура компьютеров и операционные системы**

Эргешов Атаджан НКАбд-03-23

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Самостоятельное задание . . . . .	17
<b>3</b>	<b>Выводы</b>	<b>22</b>

## Список иллюстраций

2.1	Создал каталог и файл . . . . .	6
2.2	Программа в файле lab7-1.asm . . . . .	7
2.3	Запуск программы lab7-1.asm . . . . .	8
2.4	Программа в файле lab7-1.asm . . . . .	9
2.5	Запуск программы lab7-1.asm . . . . .	10
2.6	Программа в файле lab7-1.asm . . . . .	11
2.7	Запуск программы lab7-1.asm . . . . .	12
2.8	Программа в файле lab7-2.asm . . . . .	13
2.9	Запуск программы lab7-2.asm . . . . .	14
2.10	Файл листинга lab7-2 . . . . .	15
2.11	Ошибка трансляции lab7-2 . . . . .	16
2.12	Файл листинга с ошибкой lab7-2 . . . . .	17
2.13	Программа в файле lab7-3.asm . . . . .	18
2.14	Запуск программы lab7-3.asm . . . . .	19
2.15	Программа в файле lab7-4.asm . . . . .	20
2.16	Запуск программы lab7-4.asm . . . . .	21

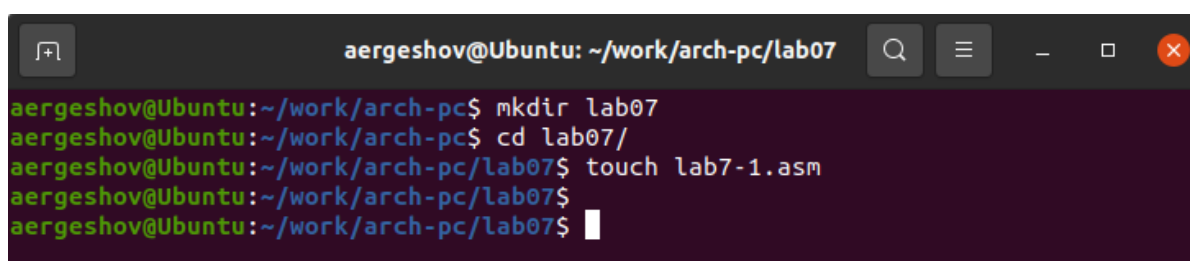
## Список таблиц

# 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

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

A screenshot of a terminal window with a dark background. The window title is 'aergeshov@Ubuntu: ~/work/arch-pc/lab07'. The terminal shows the following commands and their outputs:

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

Рис. 2.1: Создал каталог и файл

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. [2.2])

```
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 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. 2.2: Программа в файле lab7-1.asm

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

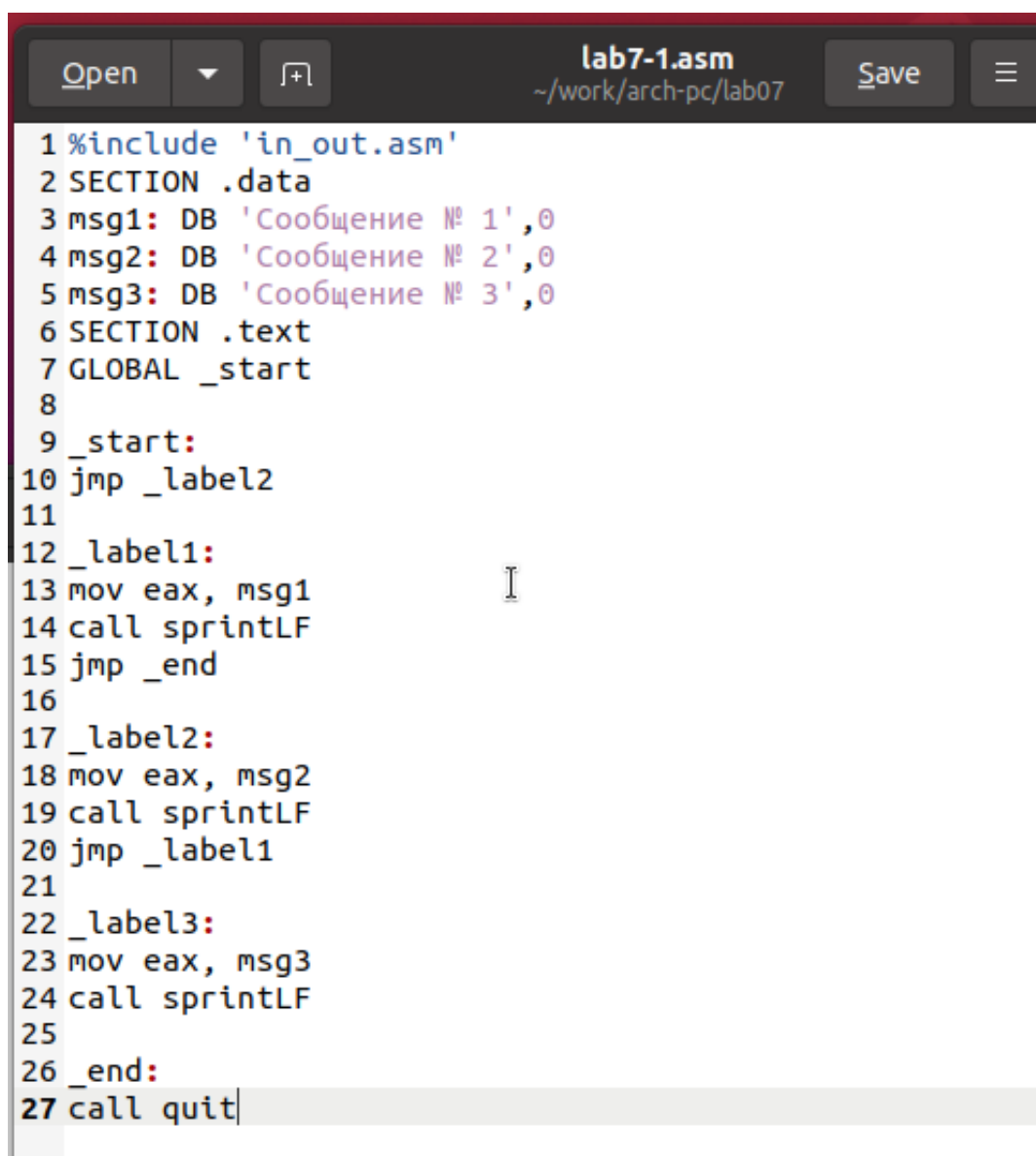
```
aergeshov@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aergeshov@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
aergeshov@Ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
aergeshov@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.3: 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. (рис. [2.4]) (рис. [2.5])





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

Рис. 2.4: Программа в файле lab7-1.asm

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

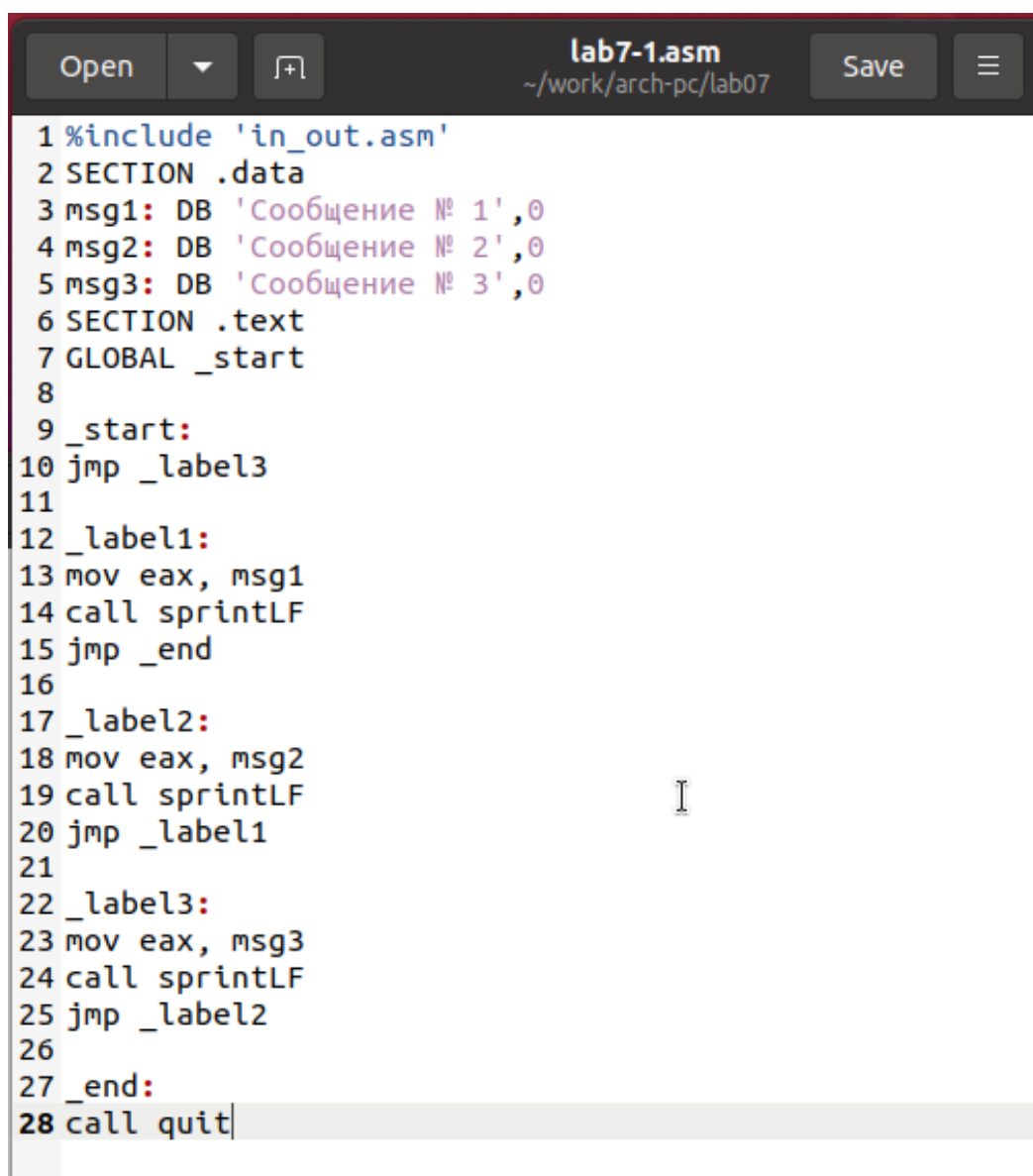
Рис. 2.5: Запуск программы lab7-1.asm

Изменил текст программы, изменив инструкции `jmp`, чтобы вывод программы был следующим (рис. [2.6]) (рис. [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 sprintfLF
15 jmp _end
16
17 _label2:
18 mov eax, msg2
19 call sprintfLF
20 jmp _label1
21
22 _label3:
23 mov eax, msg3
24 call sprintfLF
25 jmp _label2
26
27 _end:
28 call quit
```

Рис. 2.6: Программа в файле lab7-1.asm

```

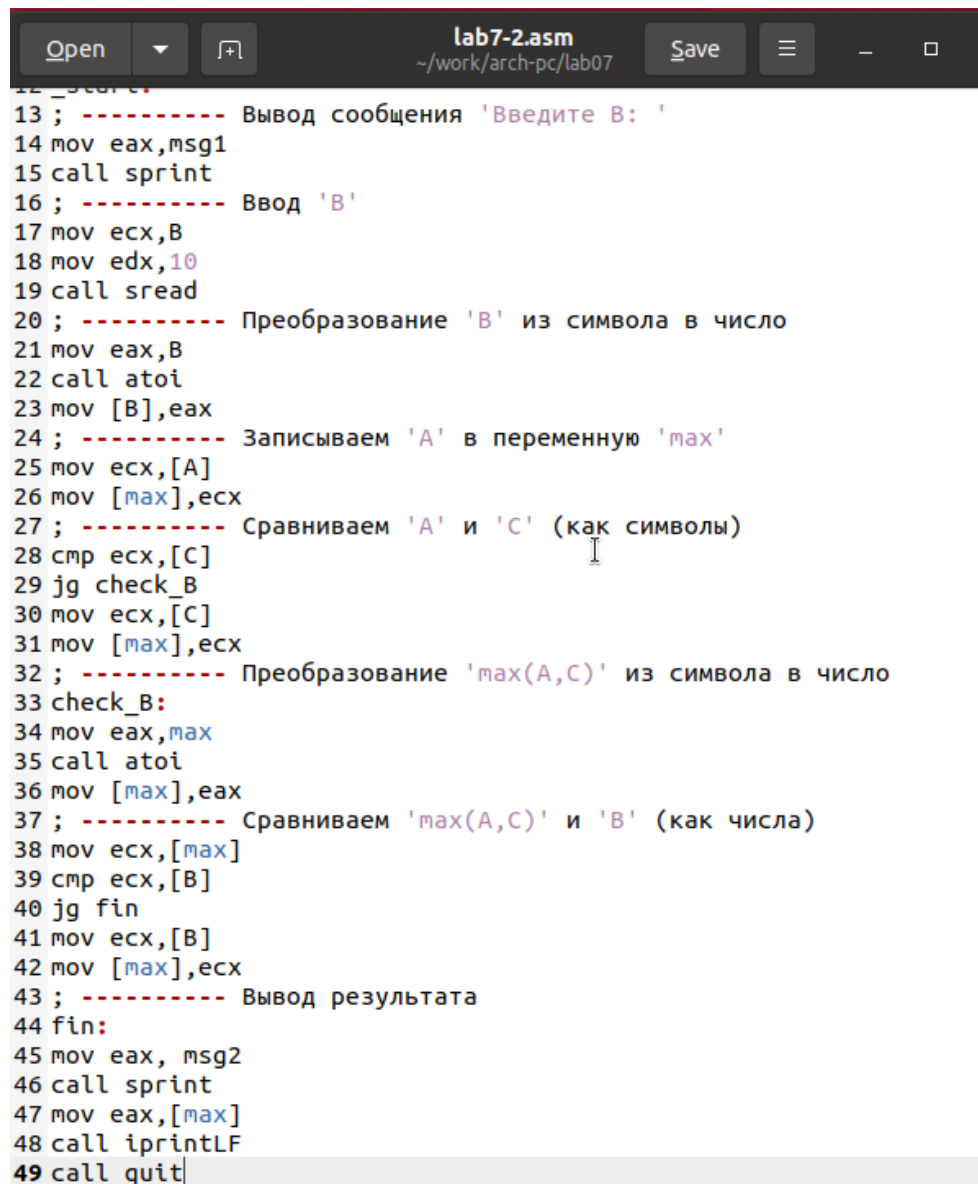
aergeshov@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aergeshov@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
aergeshov@Ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
aergeshov@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aergeshov@Ubuntu:~/work/arch-pc/lab07$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

```

Рис. 2.7: 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: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры.

Создал исполняемый файл и проверил его работу для разных значений В (рис. [2.8]) (рис. [2.9]).



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

12 ; -----
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]
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. 2.8: Программа в файле lab7-2.asm

```
aergeshov@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aergeshov@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
aergeshov@Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 10
Наибольшее число: 50
aergeshov@Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 30
Наибольшее число: 50
aergeshov@Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 60
Наибольшее число: 60
aergeshov@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.9: Запуск программы lab7-2.asm

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 (рис. [2.10])

```

180 4 0000002E 00BB00BE3A2000
181 5 00000035 32300000      A dd '20'
182 6 00000039 35300000      C dd '50'
183 7
184 8 00000000 <res 0000000A>    section .bss
185 9 0000000A <res 0000000A>    max resb 10
186 10
187 11
188 12
189 13
190 14 000000E8 B8[00000000]      B resb 10
191 15 000000ED E81DFFFFFF      section .text
192 16
193 17 000000F2 B9[0A000000]      global _start
194 18 000000F7 BA0A000000    _start:
195 19 000000FC E842FFFFFF      ; ----- Вывод сообщения 'Введите B: '
196 20
197 21 00000101 B8[0A000000]      mov eax,msg1
198 22 00000106 E891FFFFFF      call sprint
199 23 0000010B A3[0A000000]      ; ----- Ввод 'B'
200 24
201 25 00000110 8B0D[35000000]    mov ecx,B
202 26 00000116 890D[00000000]    mov edx,10
203 27
204 28 0000011C 3B0D[39000000]    call sread
205 29 00000122 7F0C
206 30 00000124 8B0D[39000000]    ; ----- Преобразование 'B' из символа в число
207 31 0000012A 890D[00000000]    mov eax,B
208 32
209 33
210 34 00000130 B8[00000000]      call atoi
211 35 00000135 E862FFFFFF      mov [B],eax
212 36 0000013A A3[00000000]      ; ----- Записываем 'A' в переменную 'max'
213 37
214 38 0000013F 8B0D[00000000]    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]

```

Рис. 2.10: Файл листинга lab7-2

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

строка 190

- 14 - номер строки в подпрограмме
- 000000E8 - адрес
- B8[00000000] - машинный код
- mov eax,msg1 - код программы - перекладывает msg1 в eax

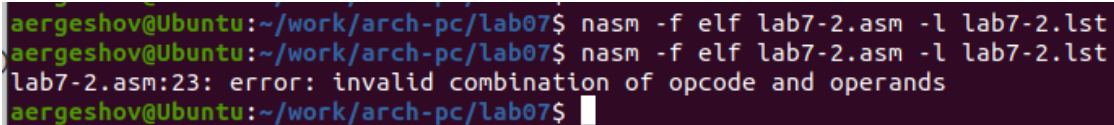
строка 191

- 15 - номер строки в подпрограмме
- 000000ED - адрес
- E81DFFFFFF - машинный код
- call sprint - код программы - вызов подпрограммы печати

строка 193

- 17 - номер строки в подпрограмме
- 000000F2 - адрес
- B9[0A000000] - машинный код
- mov esx,B - код программы - перекладывает B в esx

Открыл файл с программой lab7-2.asm и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга. (рис. [2.11]) (рис. [2.12])



```
aergeshov@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
aergeshov@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:23: error: invalid combination of opcode and operands
aergeshov@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.11: Ошибка трансляции lab7-2



```

lab7-2.asm
185 9 0000000A <res 0000000A> B resb 10
186 10 section .text
187 11 global _start
188 12 _start:
189 13 ; ----- Вывод сообщения 'Введите B: '
190 14 000000E8 B8[00000000] mov eax,msg1
191 15 000000ED E81DFFFFFF call sprint
192 16 ; ----- Ввод 'B'
193 17 000000F2 B9[0A000000] mov ecx,B
194 18 000000F7 BA0A000000 mov edx,10
195 19 000000FC E842FFFFFF call sread
196 20 ; ----- Преобразование 'B' из символа в число
197 21 00000101 B8[0A000000] mov eax,B
198 22 00000106 E891FFFFFF call atoi
199 23 mov [B],
200 23 ***** error: invalid combination of opcode and operands
201 24 ; ----- Записываем 'A' в переменную 'max'
202 25 0000010B 8B0D[35000000] mov ecx,[A]
203 26 00000111 890D[00000000] mov [max],ecx
204 27 ; ----- Сравниваем 'A' и 'C' (как символы)
205 28 00000117 3B0D[39000000] cmp ecx,[C]
206 29 0000011D 7F0C jg check_B
207 30 0000011F 8B0D[39000000] mov ecx,[C]
208 31 00000125 890D[00000000] mov [max],ecx
209 32 ; ----- Преобразование 'max(A,C)' из символа в
число
210 33 check_B:
211 34 0000012B B8[00000000] mov eax,max
212 35 00000130 E867FFFFFF call atoi
213 36 00000135 A3[00000000] mov [max],eax
214 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
215 38 0000013A 8B0D[00000000] mov ecx,[max]
216 39 00000140 3B0D[0A000000] cmp ecx,[B]
217 40 00000146 7F0C jg fin
218 41 00000148 8B0D[0A000000] mov ecx,[B]

```

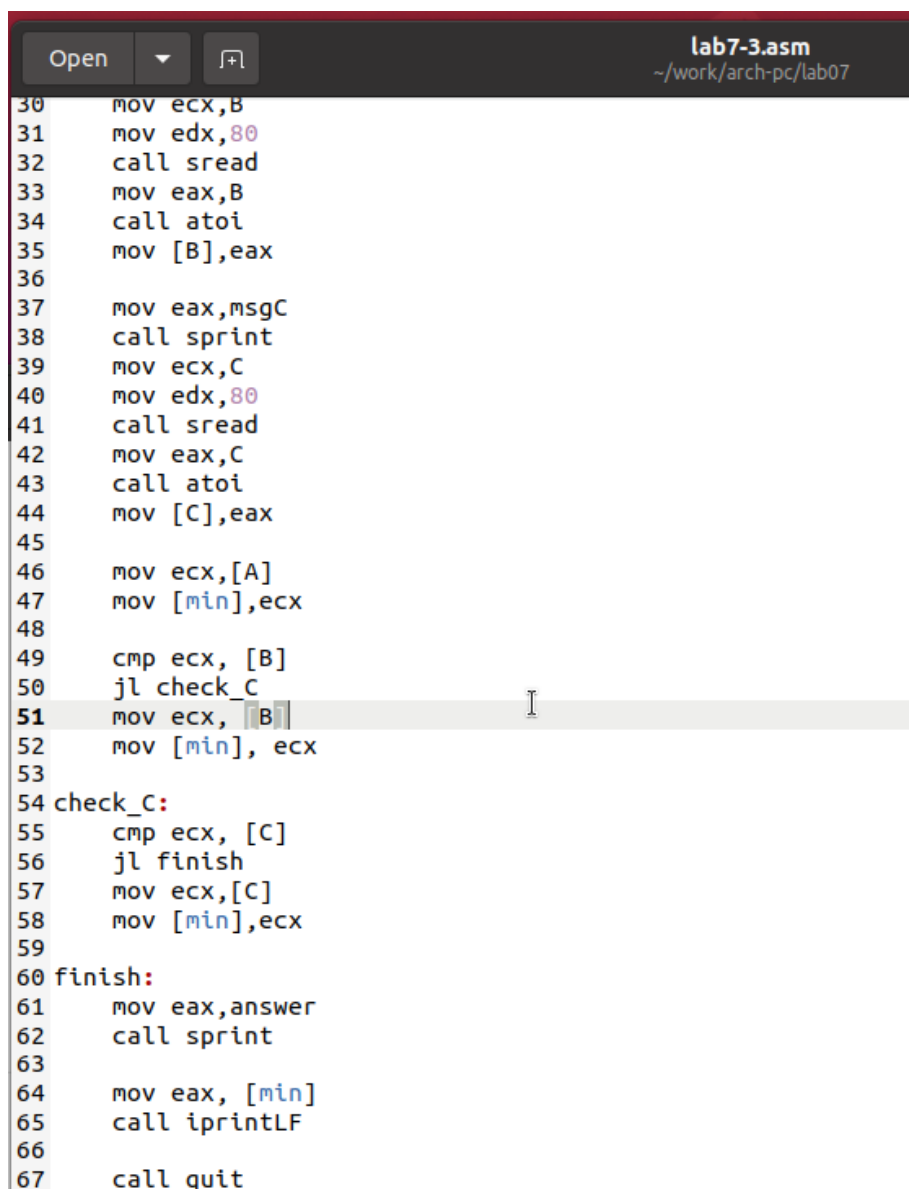
Рис. 2.12: Файл листинга с ошибкой lab7-2

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

## 2.1 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. [2.13]) (рис. [2.14])

для варианта 11 - 21,28,34



```
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
```

Рис. 2.13: Программа в файле lab7-3.asm

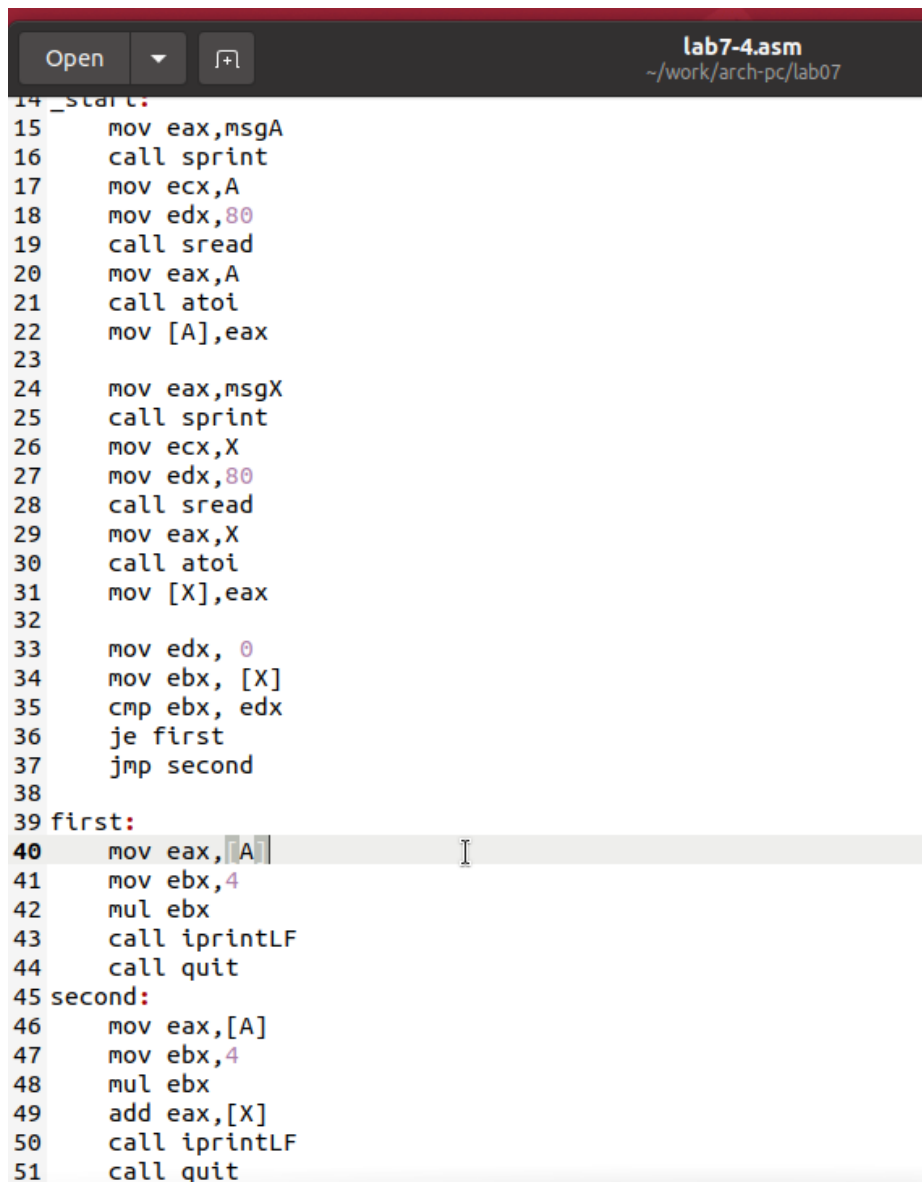
```
aergeshov@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
aergeshov@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
aergeshov@Ubuntu:~/work/arch-pc/lab07$ ./lab7-3
Input A: 21
Input B: 28
Input C: 34
Smallest: 21
aergeshov@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.14: 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. [2.15]) (рис. [2.16])

для варианта 11

$$\begin{cases} 4a, x = 0 \\ 4a + x, x \neq 0 \end{cases}$$



```
lab7-4.asm
~/work/arch-pc/lab07

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 edx, 0
34     mov ebx, [X]
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
```

Рис. 2.15: Программа в файле lab7-4.asm

```
aergeshov@Ubuntu:~/work/arch-pc/lab07$  
aergeshov@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm  
aergeshov@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4  
aergeshov@Ubuntu:~/work/arch-pc/lab07$ ./lab7-4  
Input A: 3  
Input X: 0  
12  
aergeshov@Ubuntu:~/work/arch-pc/lab07$ ./lab7-4  
Input A: 2  
Input X: 1  
9  
aergeshov@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы lab7-4.asm

## 3 Выводы

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