

# Отчёт по лабораторной работе 7

Архитектура компьютера

Довран Илиев

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	19

## Список иллюстраций

2.1	Программа lab7-1.asm . . . . .	7
2.2	Запуск программы lab7-1.asm . . . . .	8
2.3	Программа lab7-1.asm . . . . .	9
2.4	Запуск программы lab7-1.asm . . . . .	9
2.5	Программа lab7-1.asm . . . . .	10
2.6	Запуск программы lab7-1.asm . . . . .	11
2.7	Программа lab7-2.asm . . . . .	12
2.8	Запуск программы lab7-2.asm . . . . .	13
2.9	Файл листинга lab7-2 . . . . .	13
2.10	Ошибка трансляции lab7-2 . . . . .	15
2.11	Файл листинга с ошибкой lab7-2 . . . . .	15
2.12	Программа lab7-3.asm . . . . .	16
2.13	Запуск программы lab7-3.asm . . . . .	16
2.14	Программа lab7-4.asm . . . . .	18
2.15	Запуск программы lab7-4.asm . . . . .	18

## Список таблиц

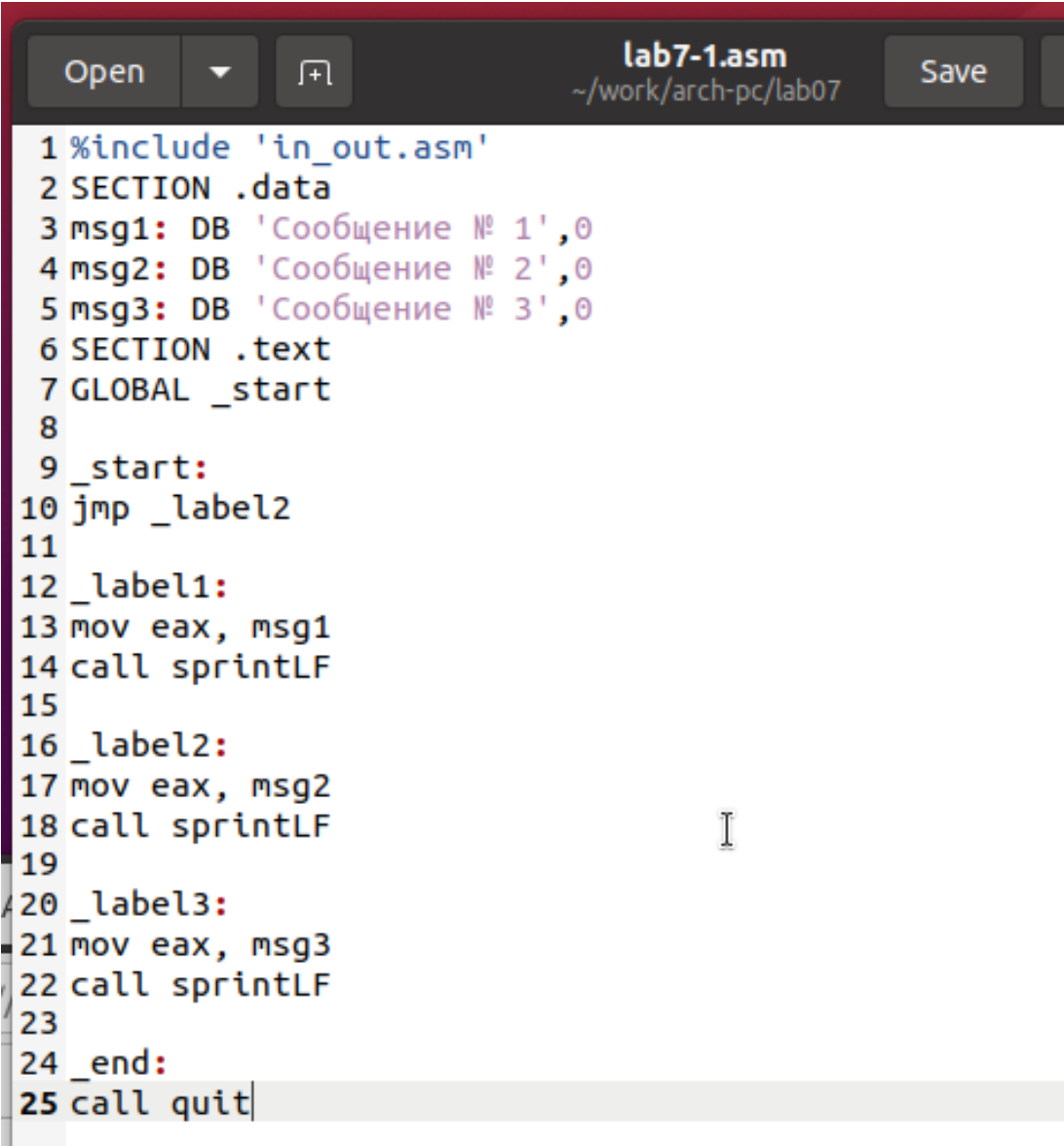
# 1 Цель работы

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

1. Сформировал папку для хранения программ лабораторной работы № 7 и создал файл lab7-1.asm в ней
2. Команда `jmp` в ассемблере NASM применяется для выполнения безусловного перехода. Проанализируем код, демонстрирующий использование команды `jmp`.

Внёс текст программы, соответствующий листингу 7.1, в файл 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
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

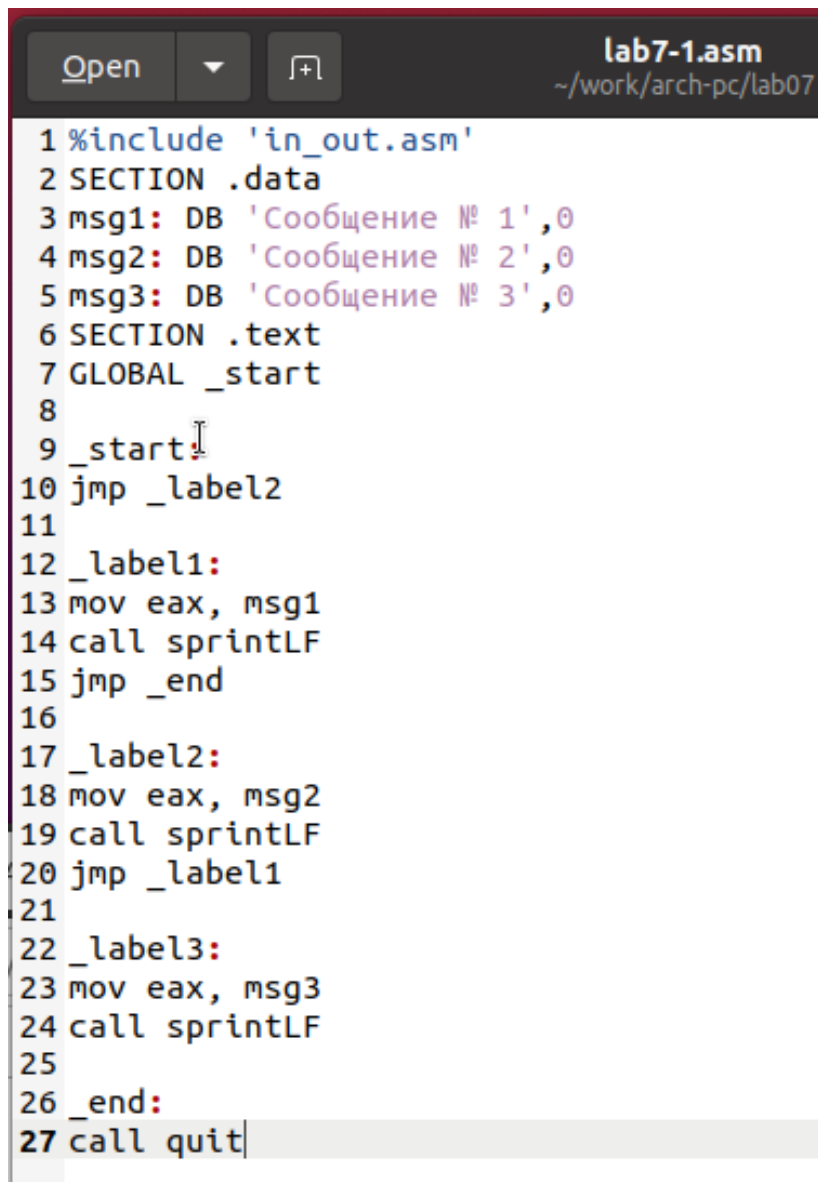
Скомпилировал программу, получил исполняемый файл и осуществил его за-  
пуск.

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

Рис. 2.2: Запуск программы lab7-1.asm

Команда `jmp` дает возможность совершать переход как вперед, так и назад по коду. Модифицируем программу так, чтобы она сначала отобразила ‘Сообщение № 2’, затем ‘Сообщение № 1’ и после этого завершила выполнение. Для достижения этого после демонстрации сообщения № 2 добавим команду `jmp` с меткой `_label1` (что означает переход к командам, выводящим сообщение № 1), и после сообщения № 1 вставим команду `jmp` с меткой `_end` (что означает переход к команде `call quit`).

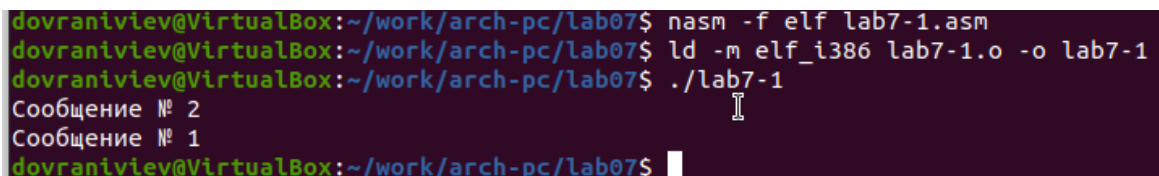
Произвёл изменения в коде программы в соответствии с листингом 7.2.



```
lab7-1.asm
~/work/arch-pc/lab07

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. 2.3: Программа lab7-1.asm



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

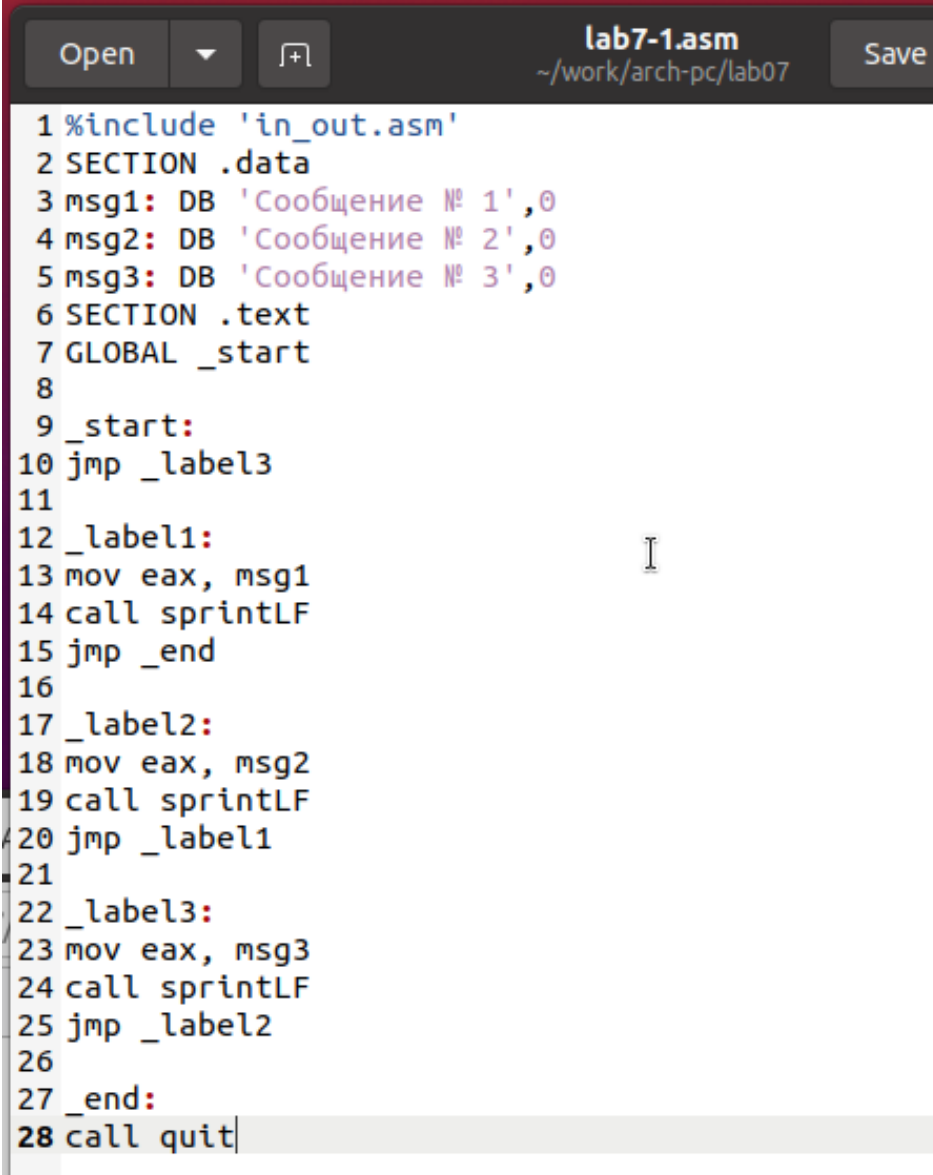
Рис. 2.4: Запуск программы lab7-1.asm

Исправил код программы, скорректировав команды `jmp` для достижения следующего порядка вывода информации:

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
```

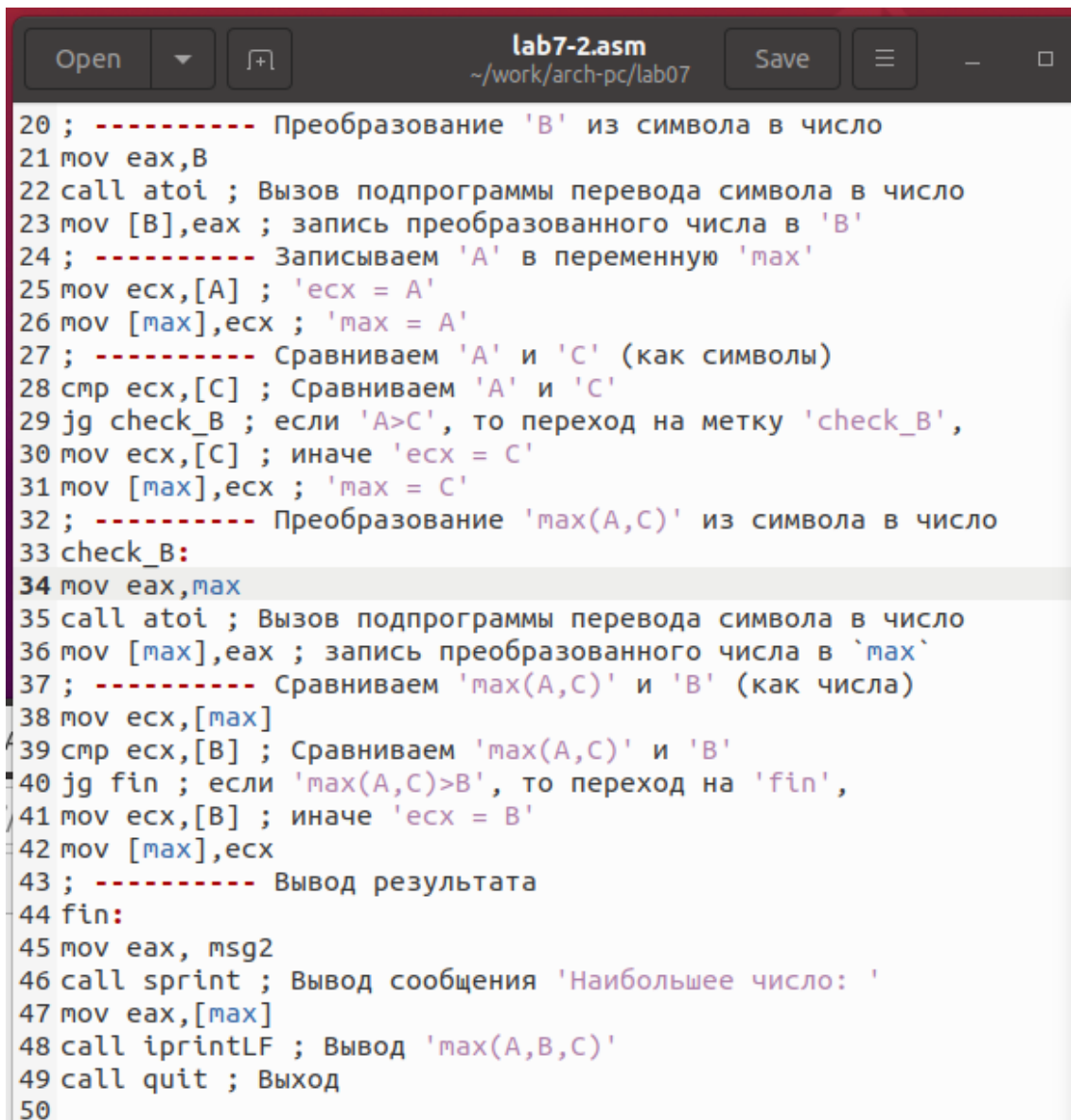
Рис. 2.5: Программа lab7-1.asm

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

Рис. 2.6: Запуск программы lab7-1.asm

3. Применение команды `jmp` всегда ведёт к совершению перехода. Однако, в процессе программирования часто возникает потребность в условных переходах, когда переход осуществляется только при выполнении определённого условия. В качестве примера возьмём программу, которая определяет и показывает на экране максимальное число из трёх целочисленных переменных: A, B и C. Значения A и C заданы в коде программы, а значение B вводится пользователем с клавиатуры.

Скомпилировал программу и осуществил проверку её работы с различными значениями B.



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

20 ; ----- Преобразование 'B' из символа в число
21 mov eax,B
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A] ; 'ecx = A'
26 mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 jg check_B ; если 'A>C', то переход на метку 'check_B',
30 mov ecx,[C] ; иначе 'ecx = C'
31 mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,max
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [max],eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 mov ecx,[max]
39 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 mov ecx,[B] ; иначе 'ecx = B'
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 mov eax, msg2
46 call sprintf ; Вывод сообщения 'Наибольшее число: '
47 mov eax,[max]
48 call iprintLF ; Вывод 'max(A,B,C)'
49 call quit ; Выход
50
```

Рис. 2.7: Программа lab7-2.asm

```
dovraniviev@VirtualBox:~/work/arch-pc/lab07$
dovraniviev@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
dovraniviev@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
dovraniviev@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 40
Наибольшее число: 50
dovraniviev@VirtualBox:~/work/arch-pc/lab07$
dovraniviev@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
dovraniviev@VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.8: Запуск программы lab7-2.asm

- В обычном режиме `nasm` генерирует только объектный файл после асемблирования. Чтобы создать файл листинга, необходимо использовать ключ `-l` и указать имя файла листинга в командной строке.

Сгенерировал файл листинга для программы, расположенной в файле `lab7-2.asm`.

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

lab7-2.asm
lab7-2.lst

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 E891FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
198 23 0000010B A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
199 24 ; ----- Записываем 'A' в переменную 'max'
200 25 00000110 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
201 26 00000116 890D[00000000] mov [max],ecx ; 'max = A'
202 27 ; ----- Сравниваем 'A' и 'C' (как символы)
203 28 0000011C 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
204 29 00000122 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B',
205 30 00000124 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
206 31 0000012A 890D[00000000] mov [max],ecx ; 'max = C'
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 ; Вызов подпрограммы перевода символа в число
211 36 0000013A A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
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] ; Сравниваем 'max(A,C)' и 'B'
215 40 0000014B 7F0C jg fin ; если 'max(A,C)>B', то переход на 'fin',
216 41 0000014D 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'
217 42 00000153 890D[00000000] mov [max],ecx
218 43 ; ----- Вывод результата
```

Рис. 2.9: Файл листинга lab7-2

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

строка 23

- 23 - номер строки
- 0000010B - адрес
- A3[0A000000] - машинный код
- mov [B],eax - код программы

строка 25

- 25 - номер строки
- 00000110 - адрес
- 8B0D[35000000] - машинный код
- mov ecx,[A]- код программы

строка 26

- 26 - номер строки
- 00000116 - адрес
- 890D[00000000] - машинный код
- mov [max],ecx - код программы

Открыл исходный файл программы lab7-2.asm и удалил один из операндов в инструкции с двумя операндами. Затем провёл ассемблирование, получив файл листинга.

```
dovraniviev@VirtualBox:~/work/arch-pc/lab07$
dovraniviev@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
dovraniviev@VirtualBox:~/work/arch-pc/lab07$
dovraniviev@VirtualBox:~/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
dovraniviev@VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.10: Ошибка трансляции lab7-2

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

lab7-2.asm
lab7-2.lst

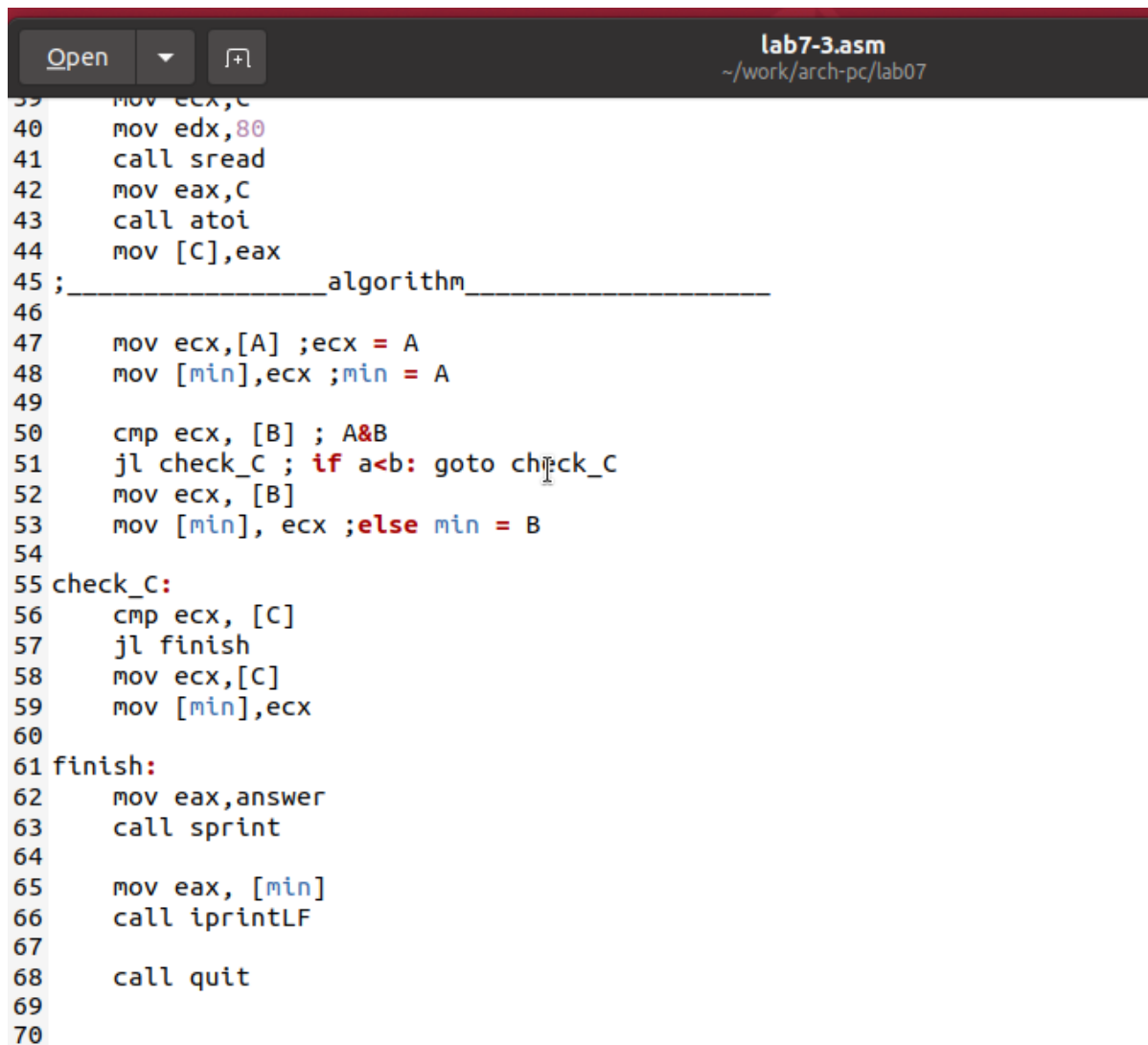
198 23 0000010B A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
199 24 ; ----- Записываем 'A' в переменную 'max'
200 25 00000110 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
201 26 00000116 890D[00000000] mov [max],ecx ; 'max = A'
202 27 ; ----- Сравниваем 'A' и 'C' (как символы)
203 28 0000011C 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
204 29 00000122 7F06 jg check_B ; если 'A>C', то переход на метку 'check_B',
205 30 mov ecx, ; иначе 'ecx = C'
206 30 *****
207 31 00000124 890D[00000000] mov [max],ecx ; 'max = C'
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 E868FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
212 36 00000134 A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
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] ; Сравниваем 'max(A,C)' и 'B'
216 40 00000145 7F0C jg fin ; если 'max(A,C)>B', то переход на 'fin',
217 41 00000147 8B0D[0A000000] mov ecx,[B] ; иначе '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 E8B2FEFFFF call sprintf ; Вывод сообщения 'Наибольшее число: '
223 47 0000015D A1[00000000] mov eax,[max]
224 48 00000162 E81FFFFF call iprintf ; Вывод 'max(A,B,C)'
225 49 00000167 E86FFFFF call quit ; Выход
226 50
```

Рис. 2.11: Файл листинга с ошибкой lab7-2

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

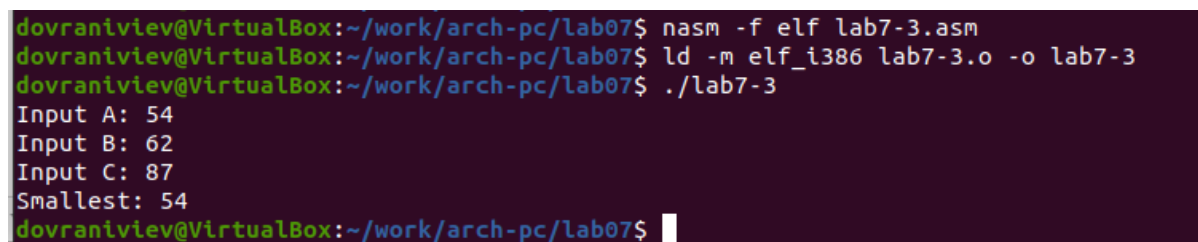
5. Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу

для варианта 5 - 54,62,87



```
39  mov ecx,C
40  mov edx,80
41  call sread
42  mov eax,C
43  call atoi
44  mov [C],eax
45 ; _____algorithm_____
46
47  mov ecx,[A] ;ecx = A
48  mov [min],ecx ;min = A
49
50  cmp ecx, [B] ; A&B
51  jl check_C ; if a<b: goto check_C
52  mov ecx, [B]
53  mov [min], ecx ;else min = B
54
55 check_C:
56  cmp ecx, [C]
57  jl finish
58  mov ecx,[C]
59  mov [min],ecx
60
61 finish:
62  mov eax,answer
63  call sprint
64
65  mov eax, [min]
66  call iprintLF
67
68  call quit
69
70
```

Рис. 2.12: Программа lab7-3.asm



```
dovraniviev@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
dovraniviev@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
dovraniviev@VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Input A: 54
Input B: 62
Input C: 87
Smallest: 54
dovraniviev@VirtualBox:~/work/arch-pc/lab07$
```

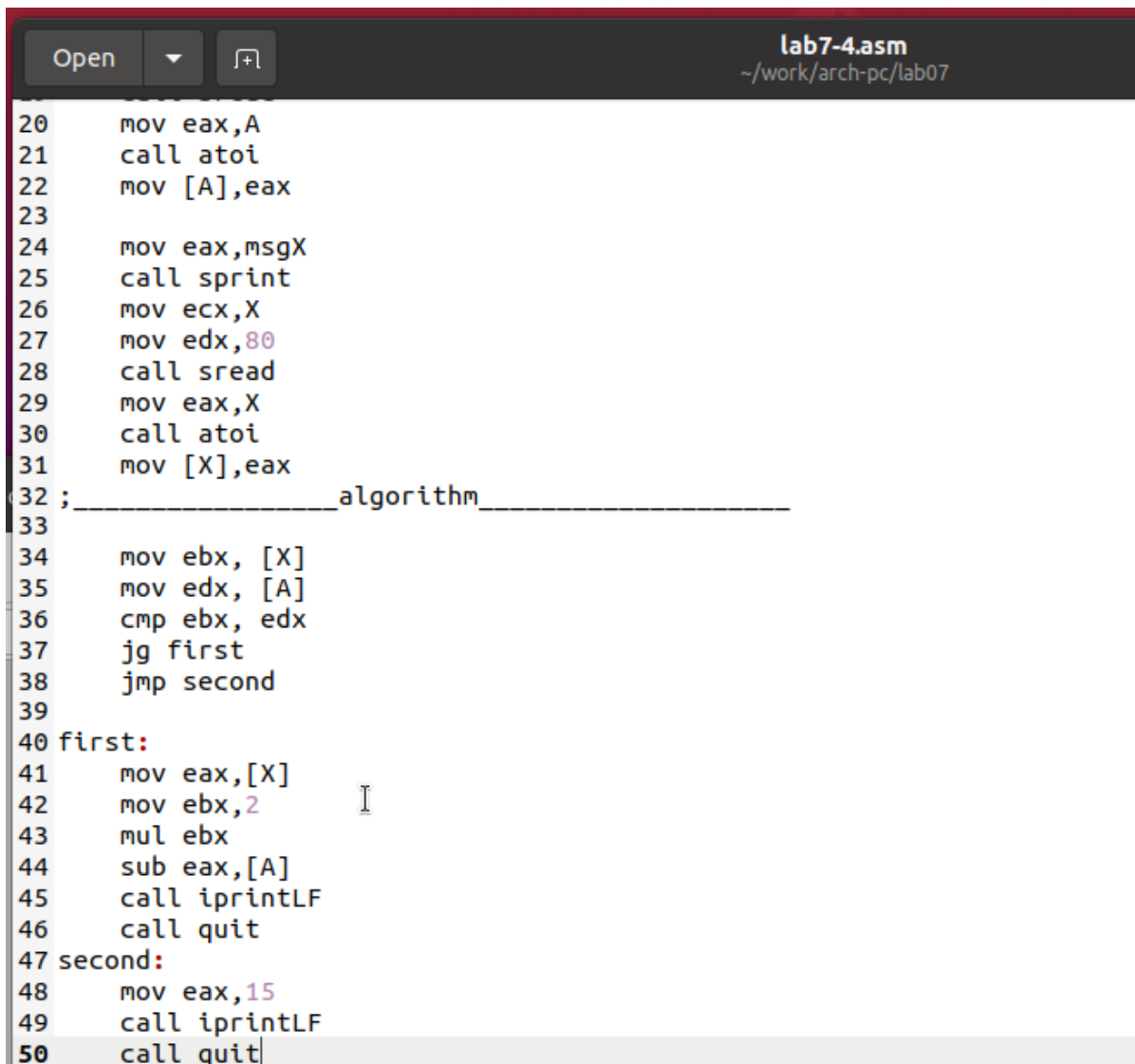
Рис. 2.13: Запуск программы lab7-3.asm

6. Напишите программу, которая для введенных с клавиатуры значений  $x$  и

а 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 5

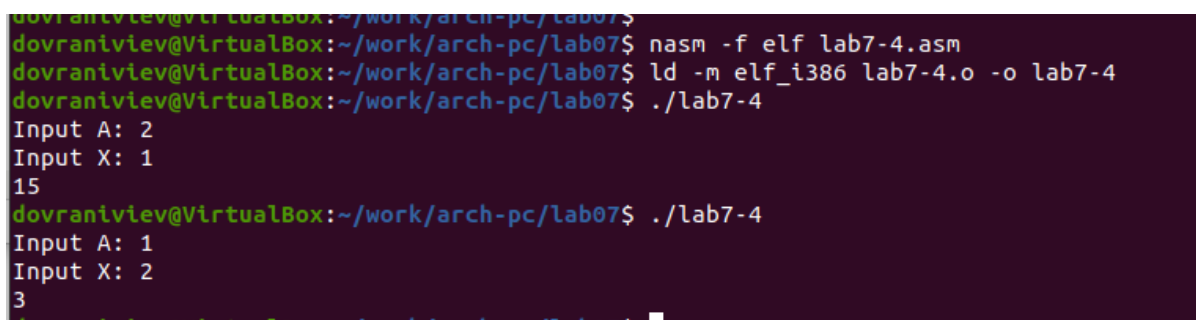
$$\begin{cases} (2x - a), x > a \\ 15, x \leq a \end{cases}$$



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

20  mov eax,A
21  call atoi
22  mov [A],eax
23
24  mov eax,msgX
25  call sprintf
26  mov ecx,X
27  mov edx,80
28  call sread
29  mov eax,X
30  call atoi
31  mov [X],eax
32 ; _____algorithm_____
33
34  mov ebx, [X]
35  mov edx, [A]
36  cmp ebx, edx
37  jg first
38  jmp second
39
40 first:
41  mov eax,[X]
42  mov ebx,2
43  mul ebx
44  sub eax,[A]
45  call iprintLF
46  call quit
47 second:
48  mov eax,15
49  call iprintLF
50  call quit
```

Рис. 2.14: Программа lab7-4.asm



```
dovraniviev@VirtualBox:~/work/arch-pc/lab07$
dovraniviev@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
dovraniviev@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4
dovraniviev@VirtualBox:~/work/arch-pc/lab07$ ./lab7-4
Input A: 2
Input X: 1
15
dovraniviev@VirtualBox:~/work/arch-pc/lab07$ ./lab7-4
Input A: 1
Input X: 2
3
```

Рис. 2.15: Запуск программы lab7-4.asm

## 3 Выводы

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