

# **Лабораторная работа №7**

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

Дмитрий Сергеевич Хохлов

# 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 . . . . .	16
<b>3</b>	<b>Выводы</b>	<b>21</b>
	<b>Список литературы</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 . . . . .	8
2.5	Запуск программы lab7-1.asm . . . . .	9
2.6	Программа в файле lab7-1.asm . . . . .	10
2.7	Запуск программы lab7-1.asm . . . . .	10
2.8	Программа в файле 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 в файле lab7-3.asm . . . . .	17
2.14	Запуск программы lab7-3.asm . . . . .	17
2.15	Программа в файле lab7-4.asm . . . . .	19
2.16	Запуск программы lab7-4.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

Сформировал директорию для хранения программ лабораторного занятия № 7 и создал файл с исходным кодом lab7-1.asm. (см. рисунок [2.1])

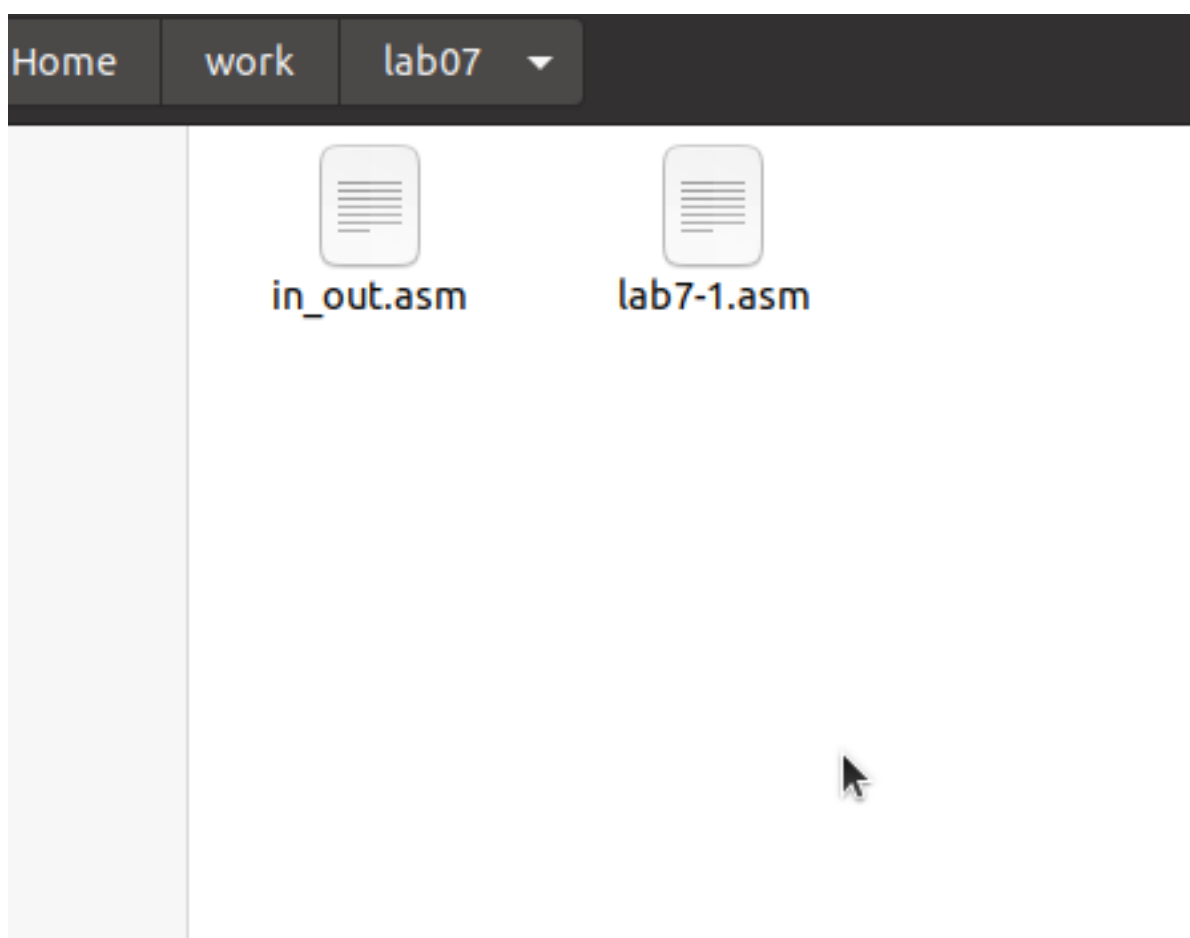
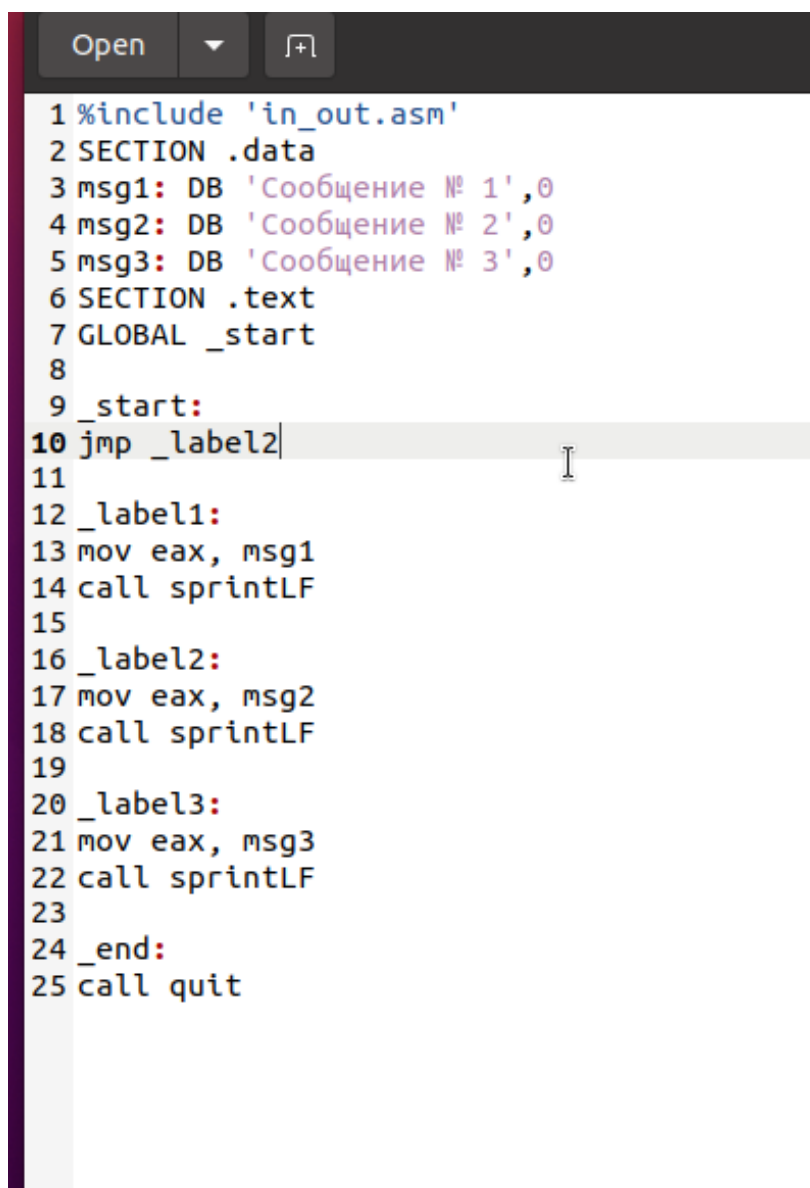


Рис. 2.1: Создал каталог и файл

Команда `jmp` в ассемблере NASM применяется для осуществления прямых

переходов. Приведем пример кода, демонстрирующего применение команды `jmp`. Внес текст примера в файл `lab7-1.asm`, оформленный как листинг 7.1. (см. рисунок [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 sprintf
15
16 _label2:
17 mov eax, msg2
18 call sprintf
19
20 _label3:
21 mov eax, msg3
22 call sprintf
23
24 _end:
25 call quit
```

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

Скомпилировал исполняемый файл и выполнил его. (см. рисунок [2.3])

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

Рис. 2.3: Запуск программы lab7-1.asm

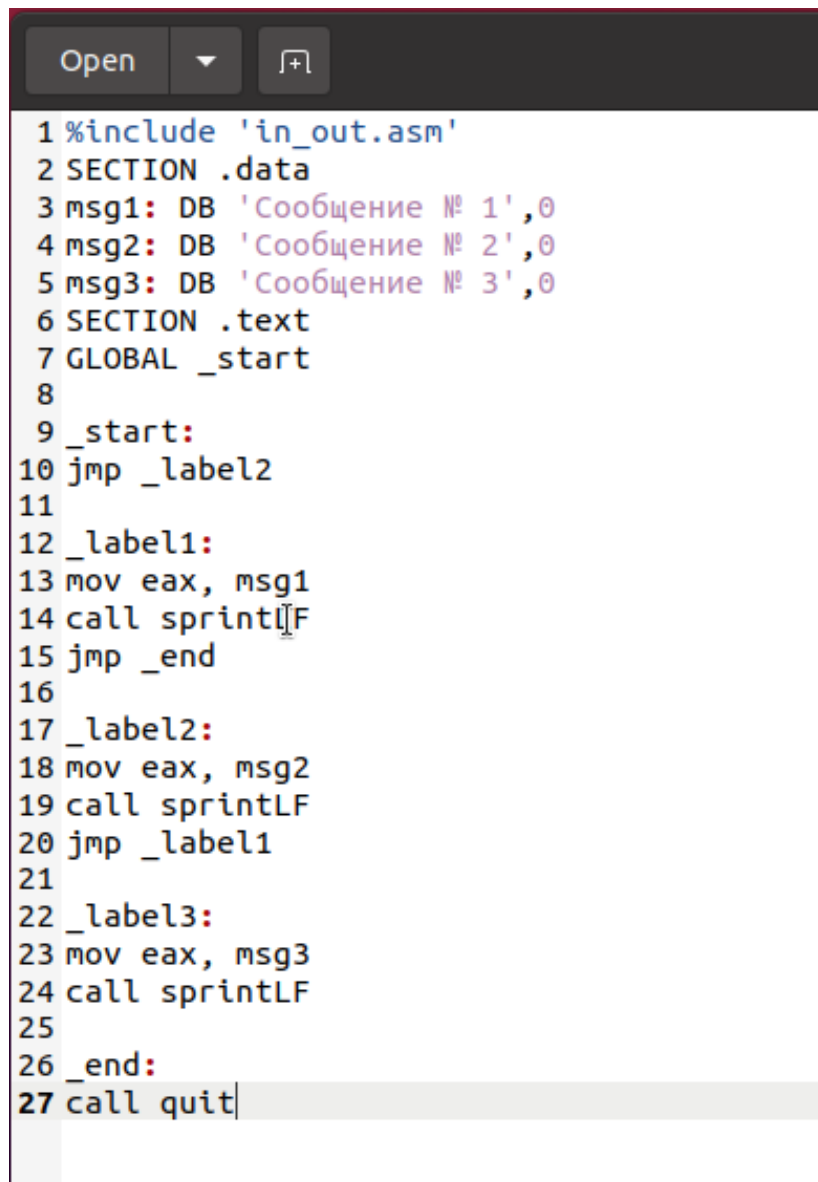
Команда `jmp` дает возможность выполнить переход как вперед, так и назад в коде. Модифицируем код так, чтобы он сначала вывел 'Сообщение № 2', затем 'Сообщение № 1' и после этого завершил выполнение. Для этого после вывода 'Сообщение № 2' вставим команду `jmp` с меткой `_label1` (то есть осуществим переход к коду, выводящему 'Сообщение № 1'), а после 'Сообщение № 1' вставим команду `jmp` с меткой `_end` (то есть перейдем к команде `call quit`).

Внес изменения в исходный код программы, следуя листингу 7.2. (см. рисунок [2.4]) (см. рисунок [2.5])

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

Рис. 2.4: Программа в файле 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 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
```

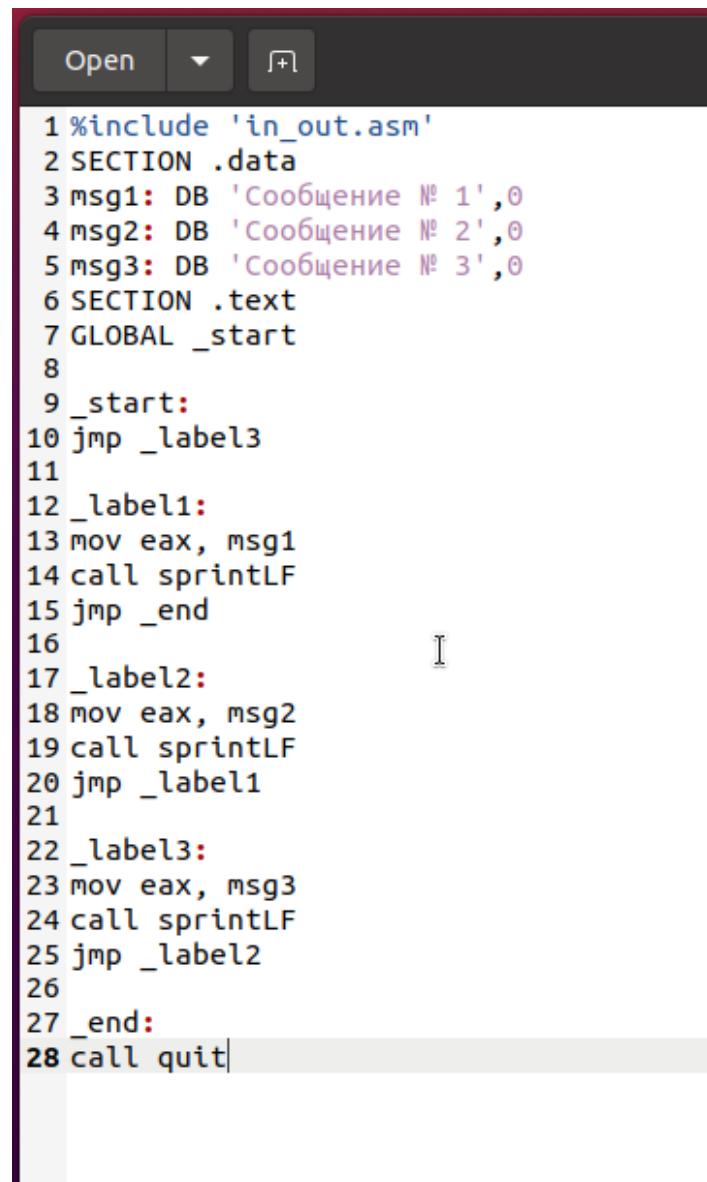
Рис. 2.5: Запуск программы lab7-1.asm

Отредактировал исходный код программы, изменив команды `jmp` для получения следующего результата выполнения (см. рисунок [2.6]) (см. рисунок [2.7])

Сообщение № 3

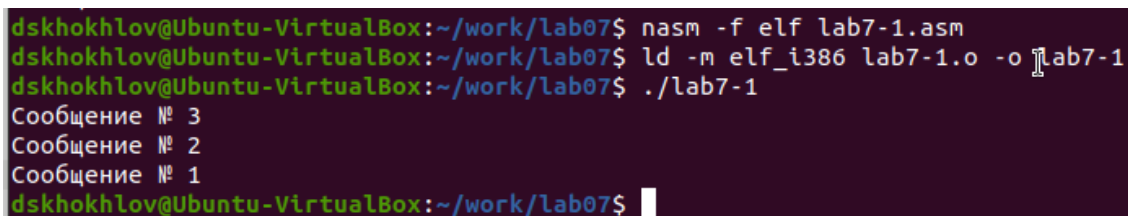
Сообщение № 2

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

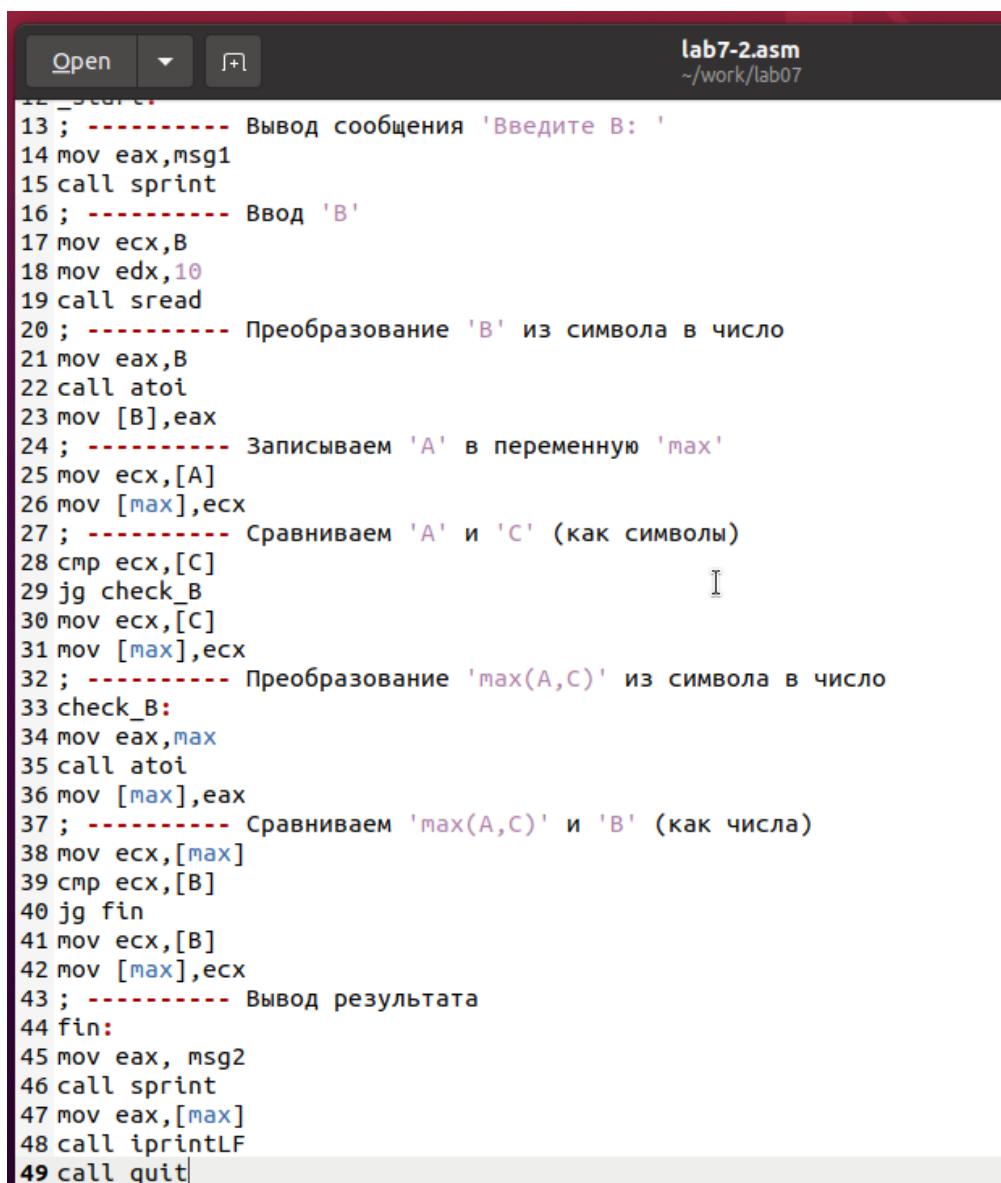


```
dskhokhlov@Ubuntu-VirtualBox:~/work/lab07$ nasm -f elf lab7-1.asm
dskhokhlov@Ubuntu-VirtualBox:~/work/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
dskhokhlov@Ubuntu-VirtualBox:~/work/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
dskhokhlov@Ubuntu-VirtualBox:~/work/lab07$
```

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.

Скомпилировал исполняемый файл и провел тестирование его функционирования с различными значениями переменной В (см. рисунок [2.8]) (см. рисунок [2.9]).



```
lab7-2.asm
~/work/lab07

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

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

Рис. 2.9: Запуск программы lab7-2.asm

Обычно при ассемблировании `nasm` создает только объектный файл. Однако можно сгенерировать файл листинга, используя ключ `-l` и указав имя файла листинга в командной строке.

Сгенерировал файл листинга для кода из файла `lab7-2.asm` (см. рисунок [2.10])

```

164 163 <1>
165 164 <1> ;----- quit -----
166 165 <1> ; Функция завершения программы
167 166 <1> quit:
168 167 000000DB BB00000000 <1> mov ebx, 0
169 168 000000E0 B801000000 <1> mov eax, 1
170 169 000000E5 CD80 <1> int 80h
171 170 000000E7 C3 <1> ret
172 2 section .data
173 3 00000000 D092D0B2D0B5D0B4D0- msg1 db 'Введите B: ',0h
174 3 00000009 B8D182D0B520423A20-
175 3 00000012 00
176 4 00000013 D09DD0B0D0B8D0B1D0- msg2 db "Наибольшее число: ",0h
177 4 0000001C BED0BBD18CD188D0B5-
178 4 00000025 D0B520D187D0B8D181-
179 4 0000002E D0BBD0BE3A2000
180 5 00000035 32300000 A dd '20'
181 6 00000039 35300000 C dd '50'
182 7 section .bss
183 8 00000000 <res 0000000A> max resb 10
184 9 0000000A <res 0000000A> B resb 10
185 10 section .text
186 11 global _start
187 12 _start:
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 E891FFFFFF call atoi
198 23 0000010B A3[0A000000] mov [B],eax
199 24 ; ----- Записываем 'A' в переменную 'max'

```

Рис. 2.10: Файл листинга lab7-2

строка 189

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

строка 190

- 15 - номер строки в подпрограмме

- 000000ED - адрес
- E81DFFFFFF - машинный код
- call sprint - код программы - вызов подпрограммы печати

строка 192

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

Открыл файл lab7-2.asm с исходным кодом и удалил один из операндов в инструкции с двумя операндами. Затем осуществил ассемблирование с созданием файла листинга. (см. рисунок [2.11]) (см. рисунок [2.12])

```
dskhokhlov@Ubuntu-VirtualBox:~/work/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
dskhokhlov@Ubuntu-VirtualBox:~/work/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:36: error: invalid combination of opcode and operands
dskhokhlov@Ubuntu-VirtualBox:~/work/lab07$
```

Рис. 2.11: Ошибка трансляции lab7-2

```

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
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                               mov [max],
212 36 *****                     error: invalid combination of opcode and operands
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
220 44                               fin:
221 45 00000154 B8[13000000]      mov eax,msg2
222 46 00000159 E8B1FEFFFF      call sprint
223 47 0000015E A1[00000000]      mov eax,[max]
224 48 00000163 E81EFFFFFF      call iprintLF
225 49 00000168 E86EFFFFFF      call quit

```

Рис. 2.12: Файл листинга с ошибкой lab7-2

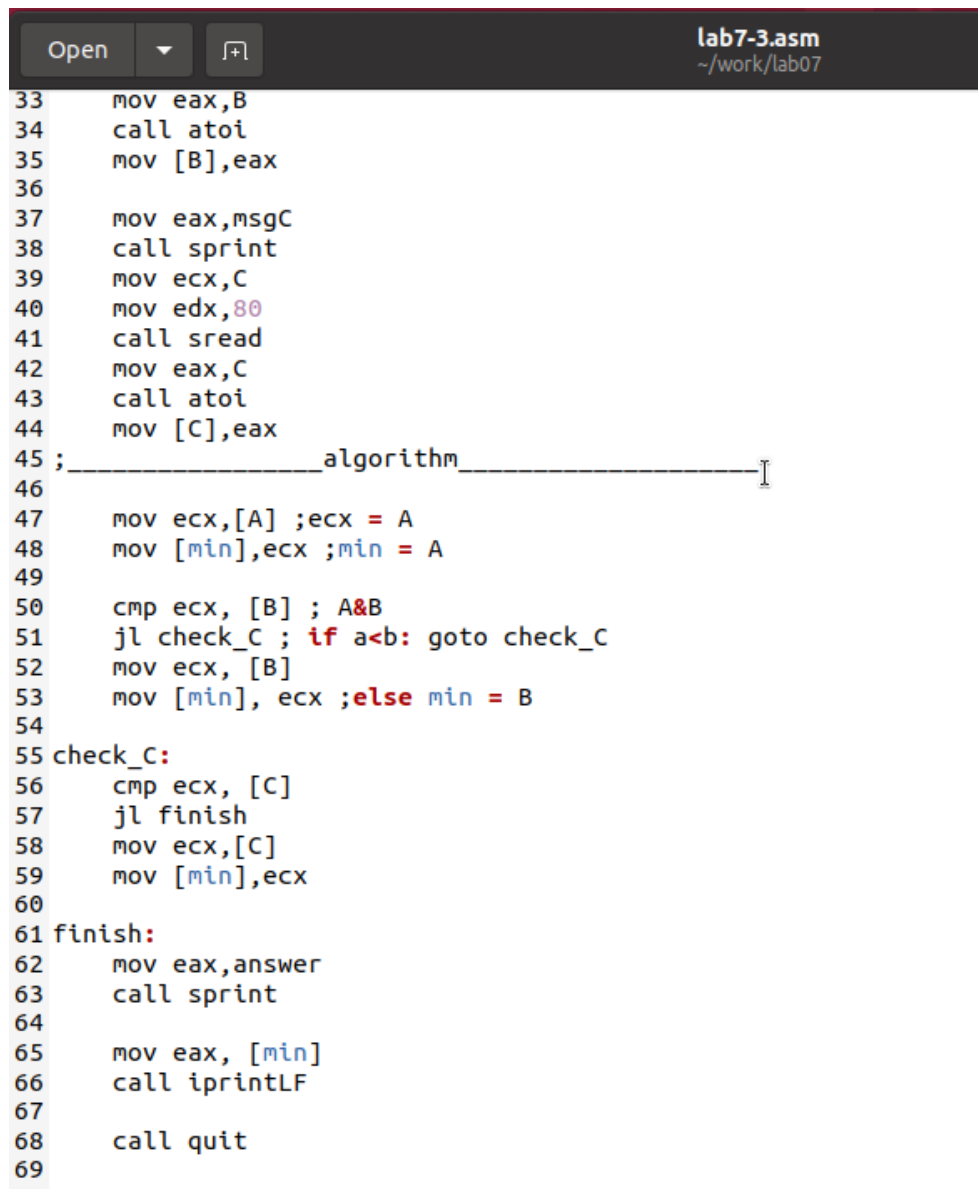
Объектный файл не был сгенерирован из-за наличия ошибки в коде. Тем не менее, был получен листинг, в котором указана ошибка.

## 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 14 - 81,22,72

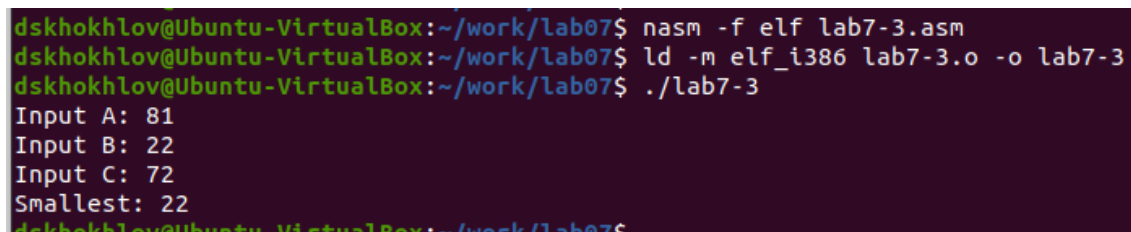




```
lab7-3.asm
~/work/lab07

33  mov eax,B
34  call atoi
35  mov [B],eax
36
37  mov eax,msgC
38  call sprintf
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 sprintf
64
65  mov eax, [min]
66  call iprintLF
67
68  call quit
69
--
```

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



```
dskhokhlov@Ubuntu-VirtualBox:~/work/lab07$ nasm -f elf lab7-3.asm
dskhokhlov@Ubuntu-VirtualBox:~/work/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
dskhokhlov@Ubuntu-VirtualBox:~/work/lab07$ ./lab7-3
Input A: 81
Input B: 22
Input C: 72
Smallest: 22
dskhokhlov@Ubuntu-VirtualBox:~/work/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 14

$$\begin{cases} 3a + 1, & x < a \\ 3x + 1, & x \geq a \end{cases}$$

Если подставить  $x = 2, a = 3$  получается 10.

Если подставить  $x = 4, a = 2$  получается 13.

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

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 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  jb first
38  jmp second
39
40 first:
41  mov eax,[A]
42  mov ebx,3
43  mul ebx
44  add eax,1
45  call iprintLF
46  call quit
47 second:
48  mov eax,[X]
49  mov ebx,3
50  mul ebx
51  add eax,1
52  call iprintLF
53  call quit
54
55
```

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

```
dskhokhlov@Ubuntu-VirtualBox:~/work/lab07$ nasm -f elf lab7-4.asm
dskhokhlov@Ubuntu-VirtualBox:~/work/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4
dskhokhlov@Ubuntu-VirtualBox:~/work/lab07$ ./lab7-4
Input A: 3
Input X: 2
10
dskhokhlov@Ubuntu-VirtualBox:~/work/lab07$ ./lab7-4
Input A: 2
Input X: 4
13
dskhokhlov@Ubuntu-VirtualBox:~/work/lab07$
```

Рис. 2.16: Запуск программы lab7-4.asm

## **3 Выводы**

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

## Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.

13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: [http://www.stolyarov.info/books/asm\\_unix](http://www.stolyarov.info/books/asm_unix).
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).