

# **Отчёт по лабораторной работе 7**

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

Асукаев Рамазан

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Теоретическое введение</b>	<b>6</b>
2.1	Команды перехода . . . . .	6
2.2	Листинг . . . . .	7
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
3.1	Реализация переходов в NASM . . . . .	8
3.2	Изучение структуры файлы листинга . . . . .	15
3.3	Задание для самостоятельной работы . . . . .	18
<b>4</b>	<b>Выводы</b>	<b>23</b>

## Список иллюстраций

3.1	Программа в файле lab7-1.asm . . . . .	9
3.2	Запуск программы lab7-1.asm . . . . .	9
3.3	Программа в файле lab7-1.asm . . . . .	11
3.4	Запуск программы lab7-1.asm . . . . .	11
3.5	Программа в файле lab7-1.asm . . . . .	12
3.6	Запуск программы lab7-1.asm . . . . .	13
3.7	Программа в файле lab7-2.asm . . . . .	14
3.8	Запуск программы lab7-2.asm . . . . .	15
3.9	Файл листинга lab7-2 . . . . .	16
3.10	Ошибка трансляции lab7-2 . . . . .	17
3.11	Файл листинга с ошибкой lab7-2 . . . . .	18
3.12	Программа в файле task7-1.asm . . . . .	19
3.13	Запуск программы task7-1.asm . . . . .	20
3.14	Программа в файле task7-2.asm . . . . .	21
3.15	Запуск программы task7-2.asm . . . . .	22

## 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 Теоретическое введение

### 2.1 Команды перехода

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

Безусловный переход выполняется инструкцией `jmp` (от англ. `jump` – прыжок), которая включает в себя адрес перехода, куда следует передать управление.

Как отмечалось выше, для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

Инструкция `cmp` является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция `cmp` является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания.

## 2.2 Листинг

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

Итак, структура листинга:

- номер строки — это номер строки файла листинга (нужно помнить, что номер строки в файле листинга может не соответствовать номеру строки в файле с исходным текстом программы);
- адрес — это смещение машинного кода от начала текущего сегмента;
- машинный код представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности. (например, инструкция `int 80h` начинается по смещению `00000020` в сегменте кода; далее идёт машинный код, в который ассемблируется инструкция, то есть инструкция `int 80h` ассемблируется в `CD80` (в шестнадцатеричном представлении); `CD80` — это инструкция на машинном языке, вызывающая прерывание ядра)
- исходный текст программы — это просто строка исходной программы вместе с комментариями (некоторые строки на языке ассемблера, например, строки, содержащие только комментарии, не генерируют никакого машинного кода, и поля «смещение» и «исходный текст программы» в таких строках отсутствуют, однако номер строки им присваивается)

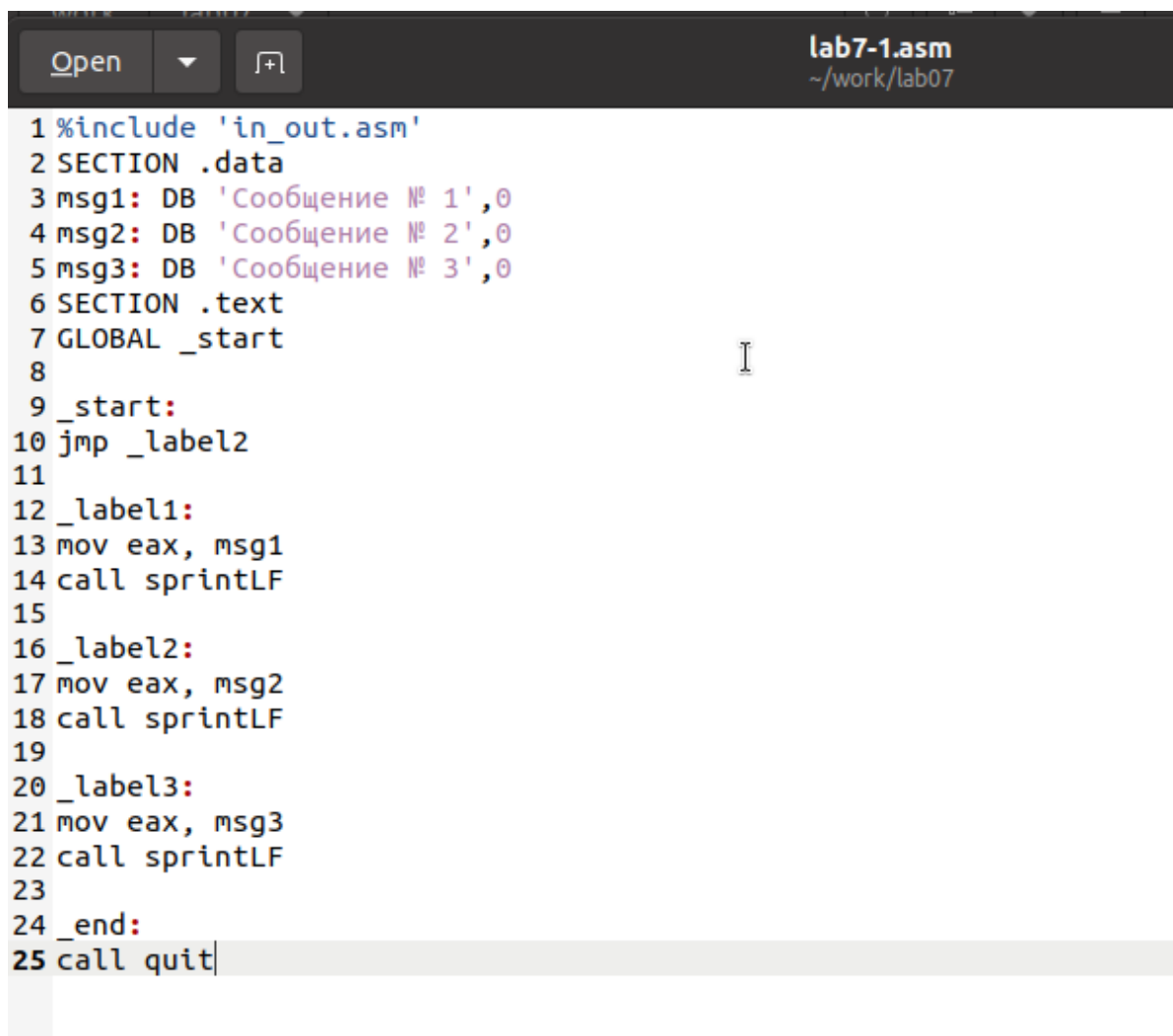
## **3 Выполнение лабораторной работы**

### **3.1 Реализация переходов в NASM**

Создал каталог для программам лабораторной работы № 7 и файл lab7-1.asm

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.



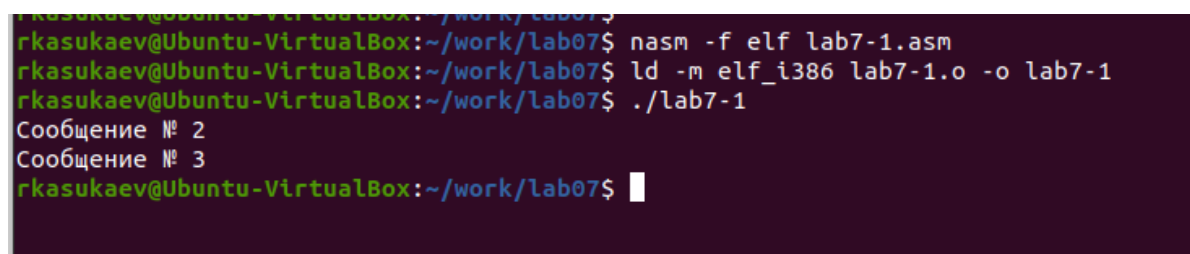


```
lab7-1.asm
~/work/lab07

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 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. 3.1: Программа в файле lab7-1.asm

Создал исполняемый файл и запустил его.

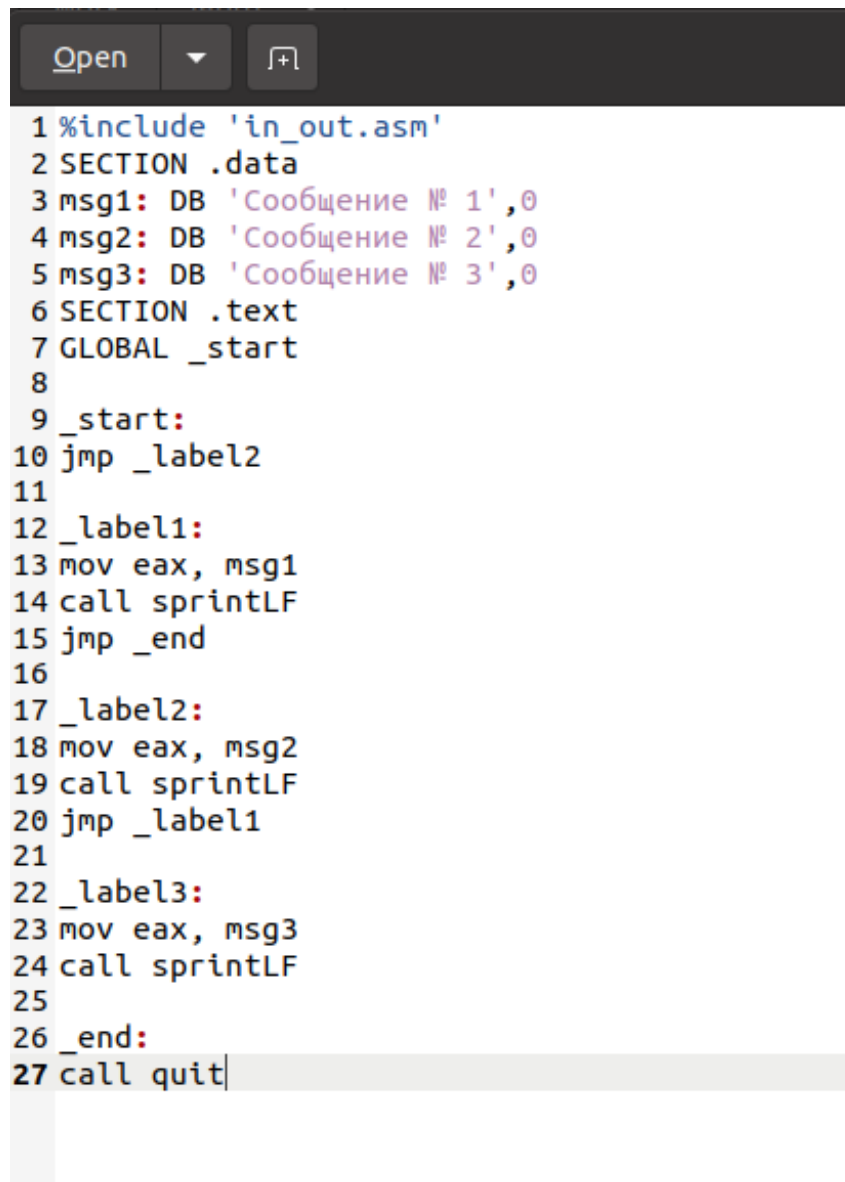


```
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$ nasm -f elf lab7-1.asm
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$
```

Рис. 3.2: Запуск программы lab7-1.asm

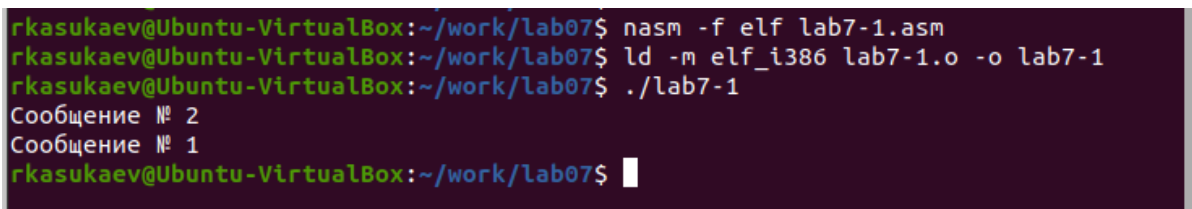
Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала 'Сообщение № 2', потом 'Сообщение № 1' и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. 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.



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



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

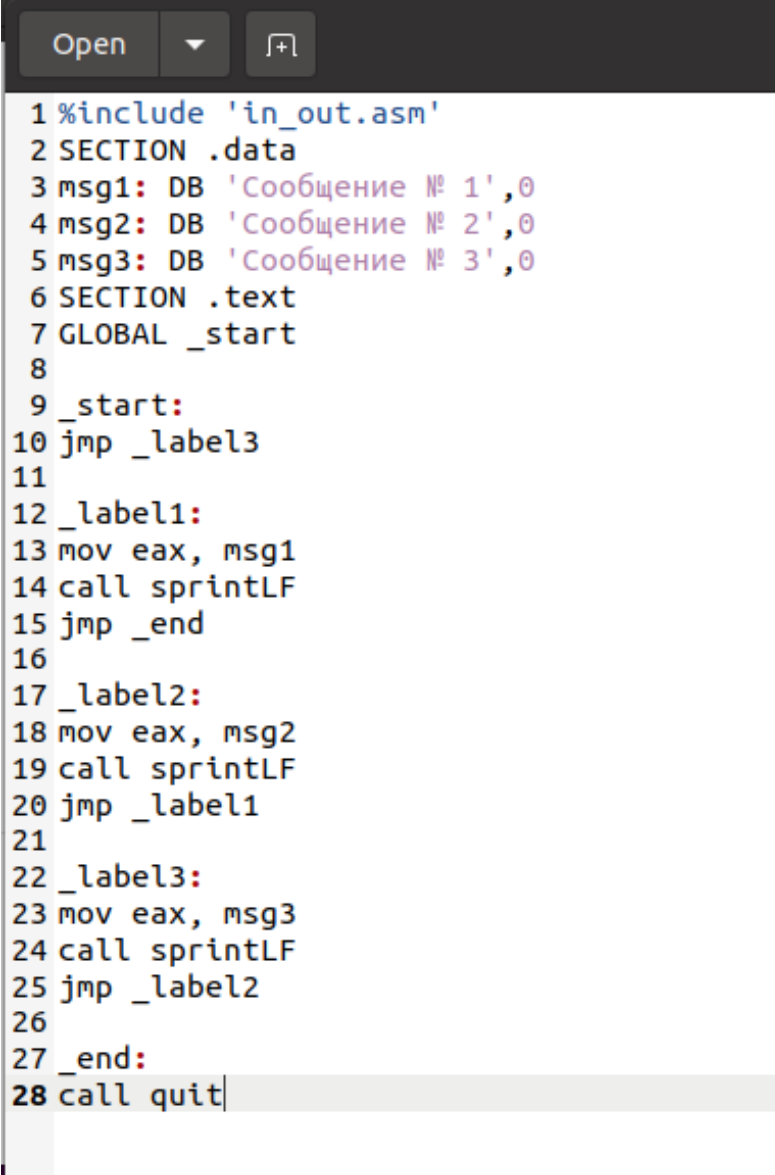
Рис. 3.4: Запуск программы lab7-1.asm

Изменил текст программы, изменив инструкции `jmp`, чтобы вывод программы был следующим:

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
```

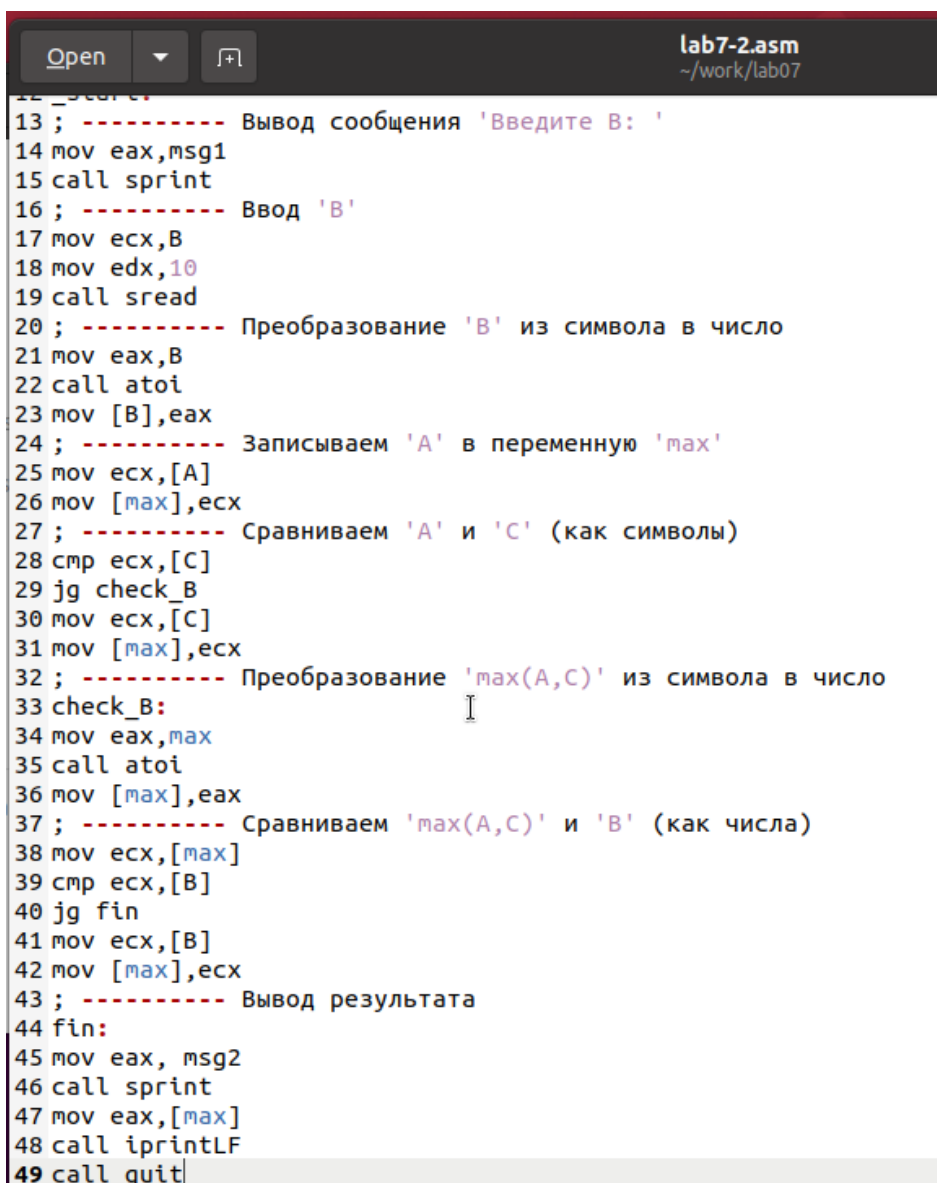
Рис. 3.5: Программа в файле lab7-1.asm

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

Рис. 3.6: 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: A, B и C. Значения для A и C задаются в программе, значение B вводится с клавиатуры.

Создал исполняемый файл и проверил его работу для разных значений B.



```
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. 3.7: Программа в файле lab7-2.asm

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

Рис. 3.8: Запуск программы lab7-2.asm

## 3.2 Изучение структуры файлы листинга

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

```
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 0000010B A3[0A000000] mov [B],eax
200 24 ; ----- Записываем 'A' в переменную 'max'
201 25 00000110 8B0D[35000000] mov ecx,[A]
202 26 00000116 890D[00000000] mov [max],ecx
203 27 ; ----- Сравниваем 'A' и 'C' (как символы)
204 28 0000011C 3B0D[39000000] cmp ecx,[C]
205 29 00000122 7F0C jg check_B
206 30 00000124 8B0D[39000000] mov ecx,[C]
207 31 0000012A 890D[00000000] mov [max],ecx
208 32 ; ----- Преобразование 'max(A,C)' из символа в число
209 33 check_B:
210 34 00000130 B8[00000000] mov eax,max
211 35 00000135 E862FFFFFF call atoi
212 36 0000013A A3[00000000] mov [max],eax
213 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
214 38 0000013F 8B0D[00000000] mov ecx,[max]
215 39 00000145 3B0D[0A000000] cmp ecx,[B]
216 40 0000014B 7F0C jg fin
217 41 0000014D 8B0D[0A000000] mov ecx,[B]
218 42 00000153 890D[00000000] mov [max],ecx
219 43 ; ----- Вывод результата
220 44 fin:
```

Рис. 3.9: Файл листинга lab7-2

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

строка 203

- 28 - номер строки в подпрограмме
- 0000011C - адрес
- 3B0D[39000000] - машинный код
- `cmp ecx,[C]` - код программы - сравнивает регистр `ecx` и переменную `C`

строка 204

- 29 - номер строки в подпрограмме
- 00000122 - адрес



- 7F0C - машинный код
- jg check\_B - код программы - если >, то переход к метке check\_B

строка 205

- 30 - номер строки в подпрограмме
- 00000124 - адрес
- 8B0D[39000000] - машинный код
- mov esx,[C] - код программы - перекладывает в регистр esx значение переменной C

Открыл файл с программой lab7-2.asm и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга.

```
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:39: error: invalid combination of opcode and operands
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$
```

Рис. 3.10: Ошибка трансляции lab7-2

```

lab7-2.asm
lab7-2.lst
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 0000010B A3[0A000000] mov [B],eax
200 24 ; ----- Записываем 'A' в переменную 'max'
201 25 00000110 8B0D[35000000] mov ecx,[A]
202 26 00000116 890D[00000000] mov [max],ecx
203 27 ; ----- Сравниваем 'A' и 'C' (как символы)
204 28 0000011C 3B0D[39000000] cmp ecx,[C]
205 29 00000122 7F0C jg check_B
206 30 00000124 8B0D[39000000] mov ecx,[C]
207 31 0000012A 890D[00000000] mov [max],ecx
208 32 ; ----- Преобразование 'max(A,C)' из символа в число
209 33 check_B:
210 34 00000130 B8[00000000] mov eax,max
211 35 00000135 E862FFFFFF call atoi
212 36 0000013A A3[00000000] mov [max],eax
213 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
214 38 0000013F 8B0D[00000000] mov ecx,[max]
215 39 cmp ecx,
216 39 *****
error: invalid combination of opcode and operands
217 40 00000145 7F0C jg fin
218 41 00000147 8B0D[0A000000] mov ecx,[B]
219 42 0000014D 890D[00000000] mov [max],ecx
220 43 ; ----- Вывод результата
221 44 fin:
222 45 00000153 B8[13000000] mov eax,msg2
223 46 00000158 E8B2FEFFFF call sprintf
224 47 0000015D A1[00000000] mov eax,[max]
225 48 00000162 E81FFFFFFF call tprintfLF
226 49 00000167 E86FFFFFFF call quit

```

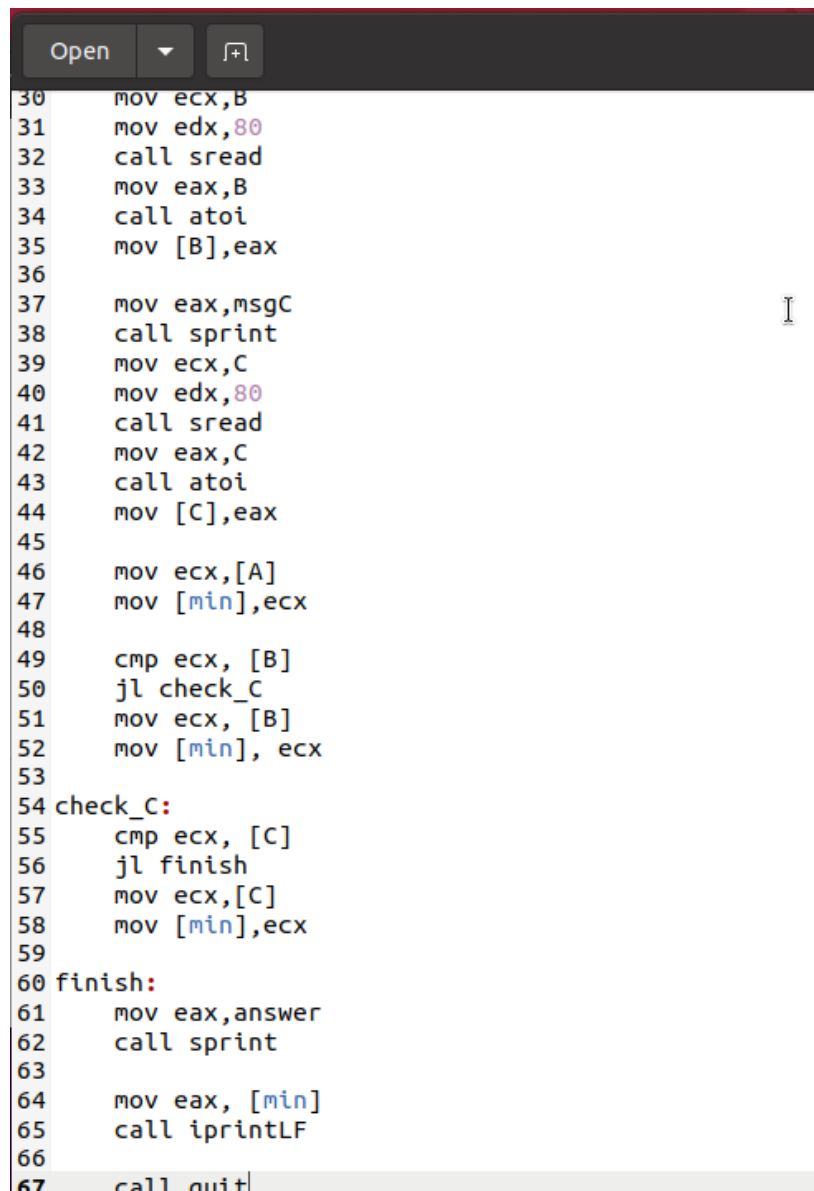
Рис. 3.11: Файл листинга с ошибкой lab7-2

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

### 3.3 Задание для самостоятельной работы

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 2 - 82,59,61



```
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. 3.12: Программа в файле task7-1.asm

```

rkasukaev@Ubuntu-VirtualBox:~/work/lab07$
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$ nasm -f elf task7-1.asm
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$ ld -m elf_i386 task7-1.o -o task7-1
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$ ./task7-1
Input A: 82
Input B: 59
Input C: 61
Smallest: 59
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$

```

Рис. 3.13: Запуск программы task7-1.asm

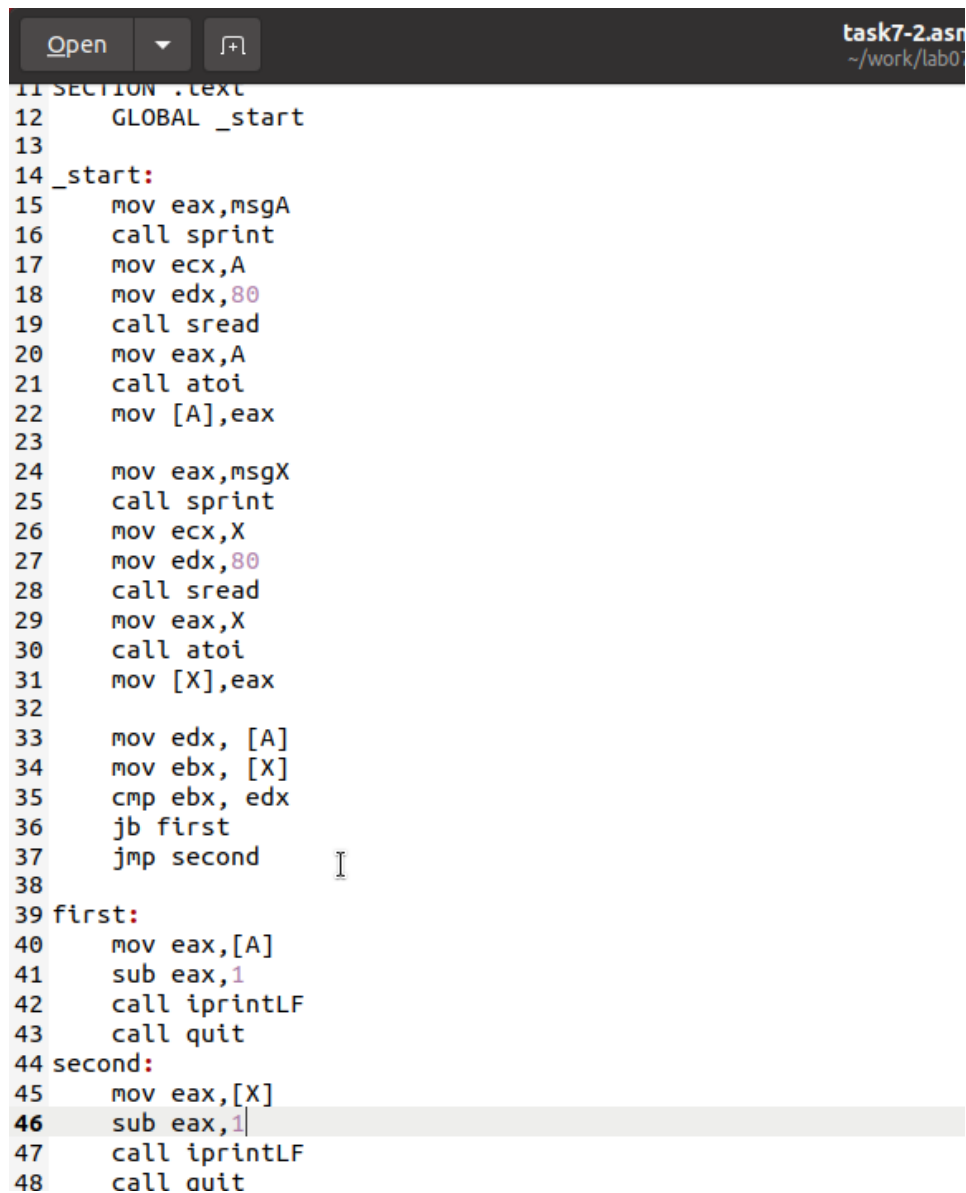
Напишите программу, которая для введенных с клавиатуры значений  $x$  и  $a$  вычисляет значение заданной функции  $f(x)$  и выводит результат вычислений. Вид функции  $f(x)$  выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений  $X$  и  $a$  из 7.6.

для варианта 2

$$\begin{cases} a - 1, x < a \\ x - 1, x \geq a \end{cases}$$

Если подставить  $x = 5, a = 7$  получается 6.

Если подставить  $x = 6, a = 4$  получается 5.



```

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 edx, [A]
34     mov ebx, [X]
35     cmp ebx, edx
36     jb first
37     jmp second
38
39 first:
40     mov eax,[A]
41     sub eax,1
42     call iprintLF
43     call quit
44 second:
45     mov eax,[X]
46     sub eax,1
47     call iprintLF
48     call quit

```

Рис. 3.14: Программа в файле task7-2.asm

```
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$  
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$ nasm -f elf task7-2.asm  
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$ ld -m elf_i386 task7-2.o -o task7-2  
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$ ./task7-2  
Input A: 7  
Input X: 5  
6  
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$ ./task7-2  
Input A: 4  
Input X: 6  
5  
rkasukaev@Ubuntu-VirtualBox:~/work/lab07$
```

Рис. 3.15: Запуск программы task7-2.asm

## 4 Выводы

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