

# **Отчёт по лабораторной работе 7**

**Архитектура компьютеров**

Хзиба Хаким НПИбд-02-24

# 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 . . . . .	14
<b>3</b>	<b>Выводы</b>	<b>19</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 . . . . .	7
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 . . . . .	11
2.9	Запуск программы lab7-2.asm . . . . .	12
2.10	Файл листинга lab7-2 . . . . .	12
2.11	Ошибка трансляции lab7-2 . . . . .	13
2.12	Файл листинга с ошибкой lab7-2 . . . . .	14
2.13	Программа в файле prog1.asm . . . . .	15
2.14	Запуск программы prog1.asm . . . . .	15
2.15	Программа в файле prog2.asm . . . . .	17
2.16	Запуск программы prog2.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

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

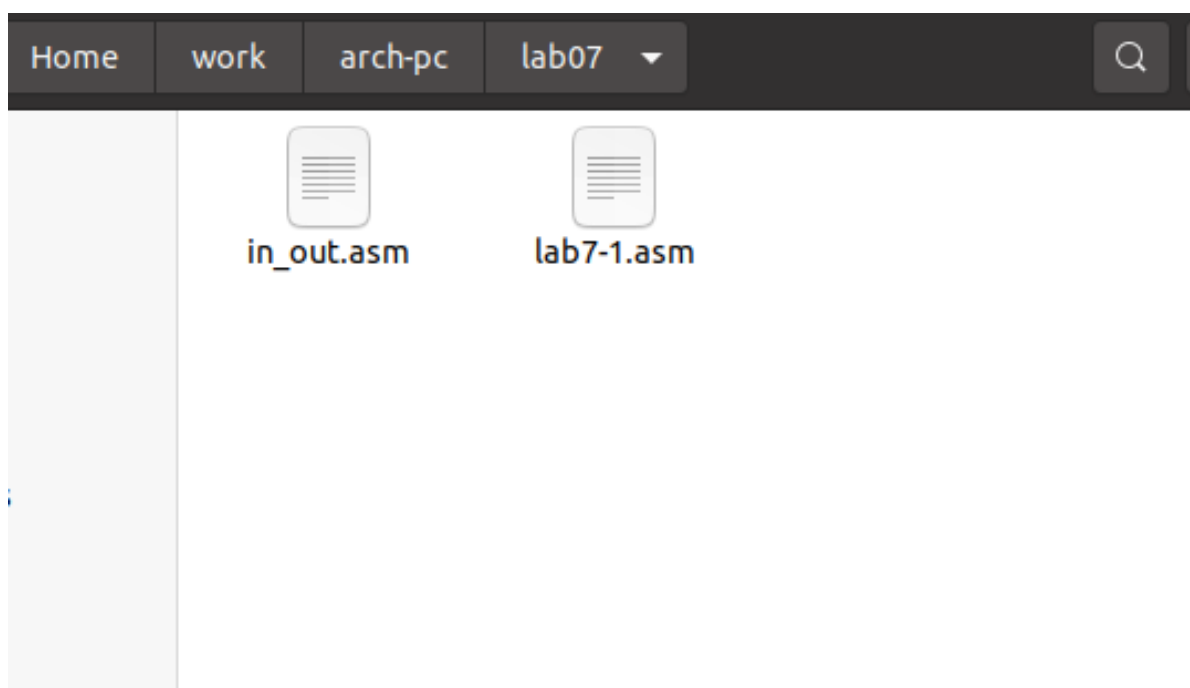


Рис. 2.1: Создал каталог и файл

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)

```

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
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

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

```

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

```
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 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



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

```

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 _label3
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 jmp _label2
26
27 _end:
28 call quit

```

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

```

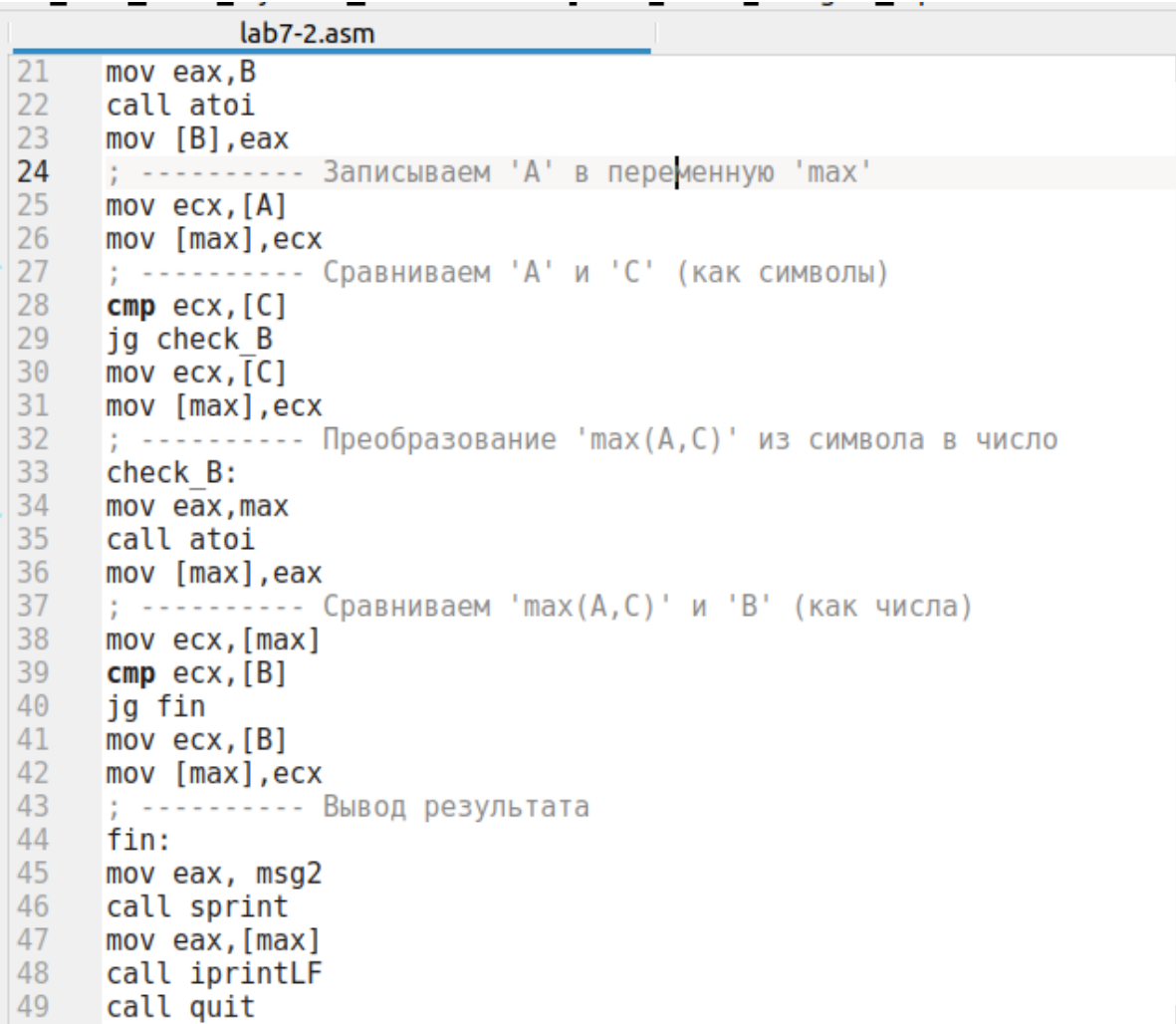
hakimkh@vm-pc:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
hakimkh@vm-pc:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
hakimkh@vm-pc:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
hakimkh@vm-pc:~/work/arch-pc/lab07$

```

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: 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 (рис. 2.8) (рис. 2.9).



```
lab7-2.asm
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

```

hakimkh@vm-pc:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
hakimkh@vm-pc:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
hakimkh@vm-pc:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
hakimkh@vm-pc:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 80
Наибольшее число: 80
hakimkh@vm-pc:~/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)

	lab7-2.lst	lab7-2.asm
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'
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]

Рис. 2.10: Файл листинга lab7-2

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

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 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)

```
hakimkh@vm-pc:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
hakimkh@vm-pc:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:39: error: invalid combination of opcode and operands
hakimkh@vm-pc:~/work/arch-pc/lab07$
```

Рис. 2.11: Ошибка трансляции lab7-2

	lab7-2.lst	lab7-2.asm
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	cmp ecx,
215	39 *****	error: invalid combination of opcode and operands
216	40 00000145 7F0C	jg fin
217	41 00000147 8B0D[0A000000]	mov 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 E8B2FFFFFF	call sprint
223	47 0000015D A1[00000000]	mov eax,[max]
224	48 00000162 E81FFFFFFF	call iprintLF
225	49 00000167 E86FFFFFFF	call quit
226		

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 2 - 82,59,61

prog1.asm	
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

Рис. 2.13: Программа в файле prog1.asm

```

hakimkh@vm-pc:~/work/arch-pc/lab07$
hakimkh@vm-pc:~/work/arch-pc/lab07$ nasm -f elf prog1.asm
hakimkh@vm-pc:~/work/arch-pc/lab07$ ld -m elf_i386 prog1.o -o prog1
hakimkh@vm-pc:~/work/arch-pc/lab07$ ./prog1
Input A: 82
Input B: 59
Input C: 61
Smallest: 59
hakimkh@vm-pc:~/work/arch-pc/lab07$

```

Рис. 2.14: Запуск программы prog1.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)

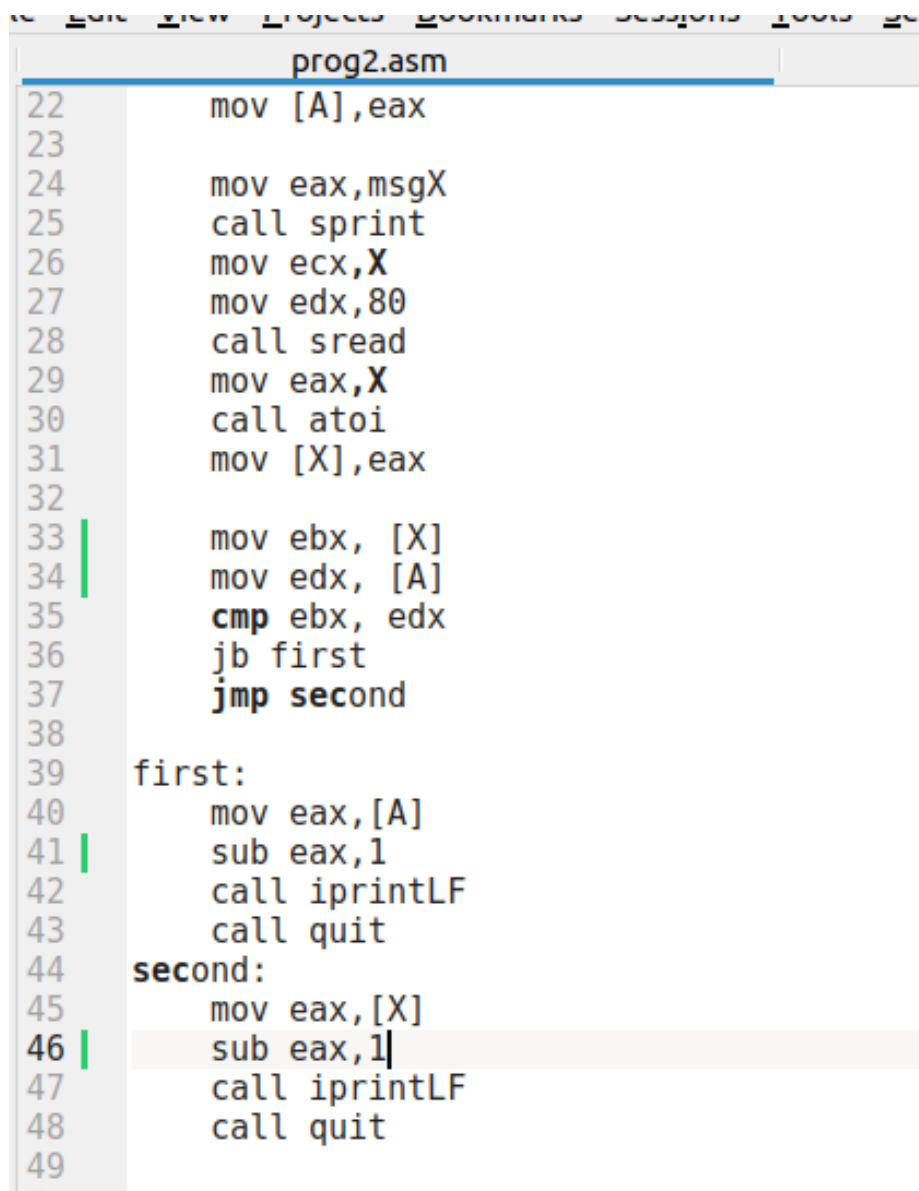
для варианта 2

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

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

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





```
prog2.asm
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     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
49
50
```

Рис. 2.15: Программа в файле prog2.asm

```
hakimkh@vm-pc:~/work/arch-pc/lab07$  
hakimkh@vm-pc:~/work/arch-pc/lab07$ nasm -f elf prog2.asm  
hakimkh@vm-pc:~/work/arch-pc/lab07$ ld -m elf_i386 prog2.o -o prog2  
hakimkh@vm-pc:~/work/arch-pc/lab07$ ./prog2  
Input A: 7  
Input X: 5  
6  
hakimkh@vm-pc:~/work/arch-pc/lab07$ ./prog2  
Input A: 4  
Input X: 6  
5  
hakimkh@vm-pc:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы prog2.asm

## 3 Выводы

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