

Отчёт по лабораторной работе 7

Архитектура компьютеров

Хиджази Мохамад

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Самостоятельное задание	15
3	Выводы	20

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	12
2.9	Запуск программы lab7-2.asm	12
2.10	Файл листинга lab7-2	13
2.11	Ошибка трансляции lab7-2	14
2.12	Файл листинга с ошибкой lab7-2	15
2.13	Программа в файле prog1.asm	16
2.14	Запуск программы prog1.asm	16
2.15	Программа в файле prog2.asm	18
2.16	Запуск программы prog2.asm	19

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)

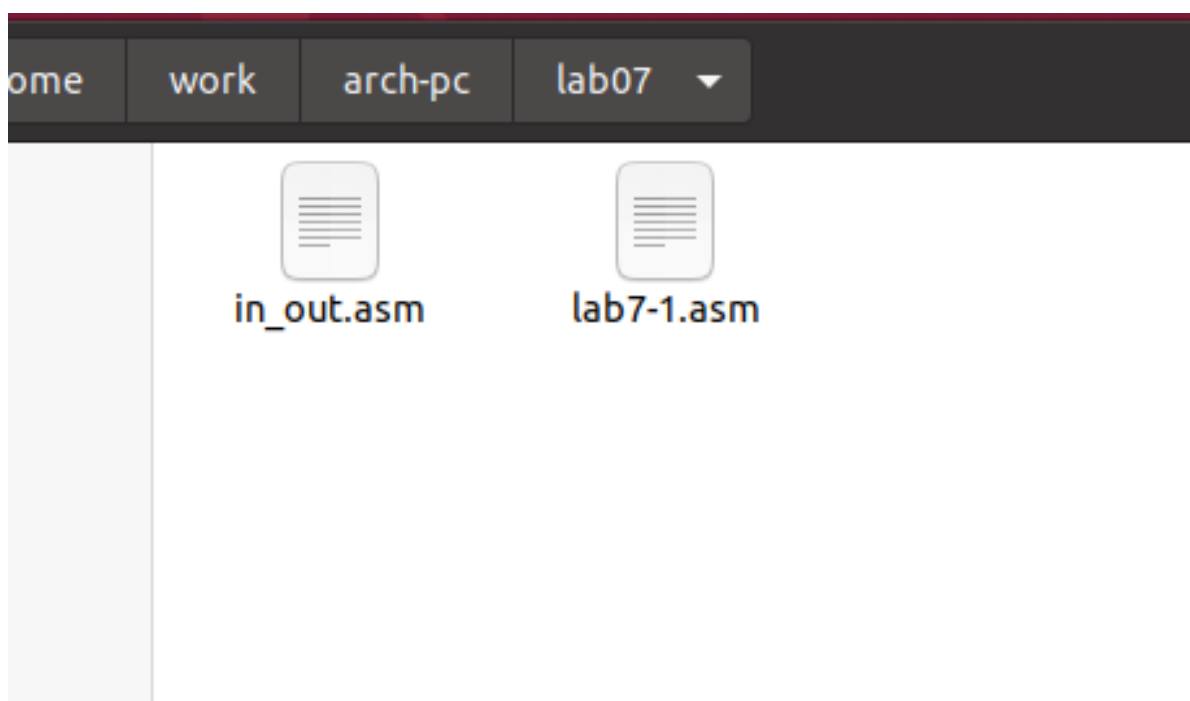
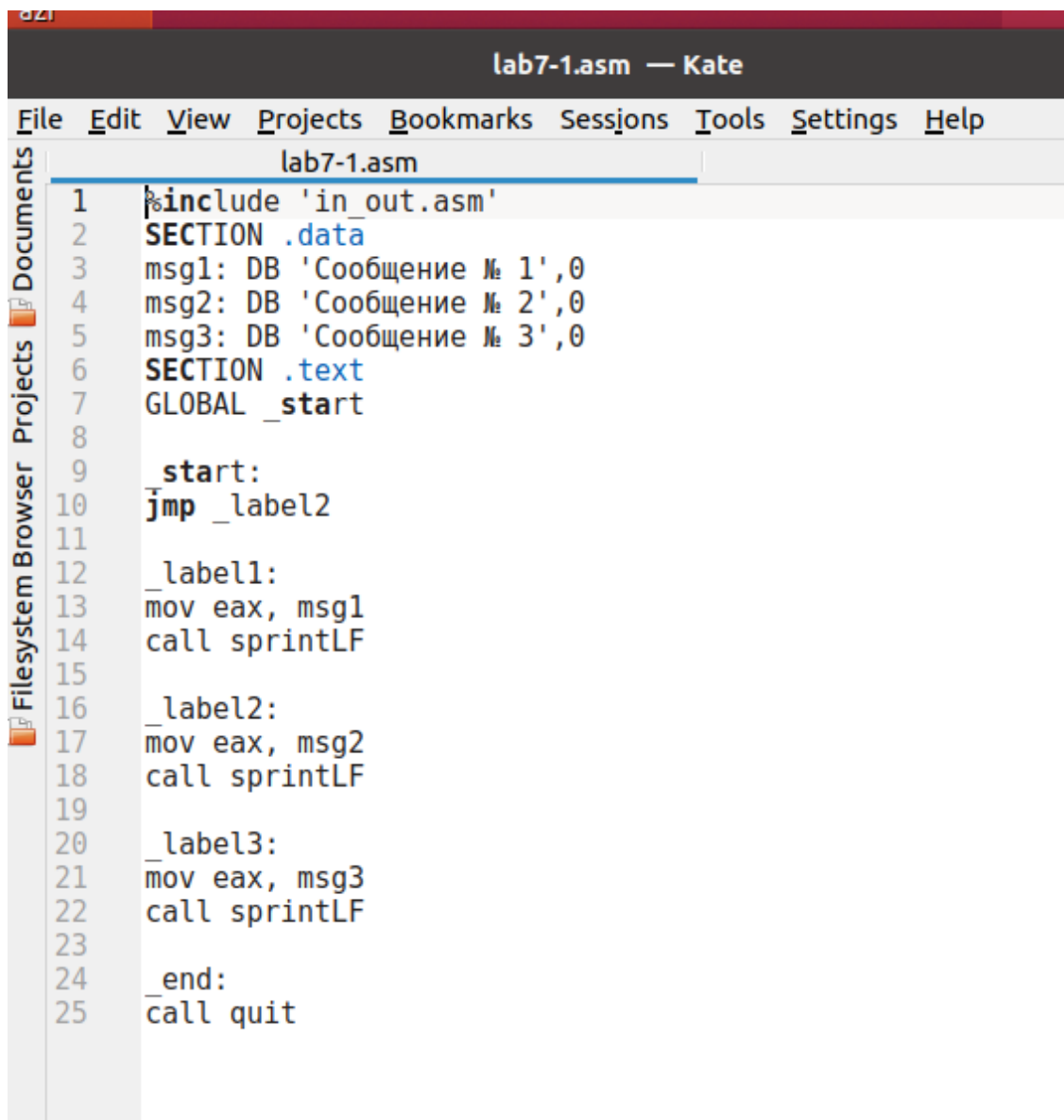


Рис. 2.1: Создал каталог и файл

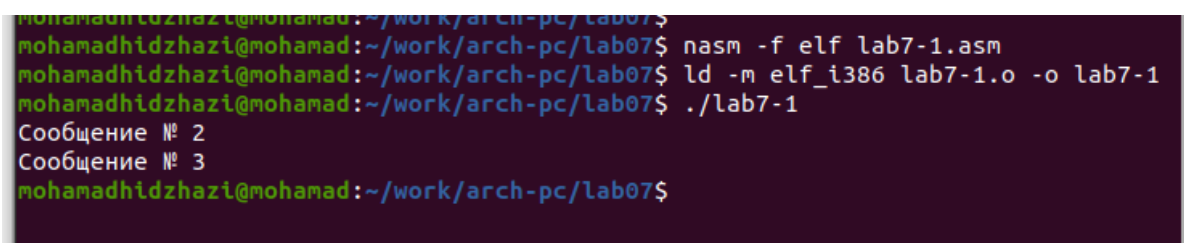
Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Написал в файл lab7-1.asm текст программы из листинга 7.1. (рис. 2.2)



```
lab7-1.asm — Kate
File Edit View Projects Bookmarks Sessions Tools Settings Help
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
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)

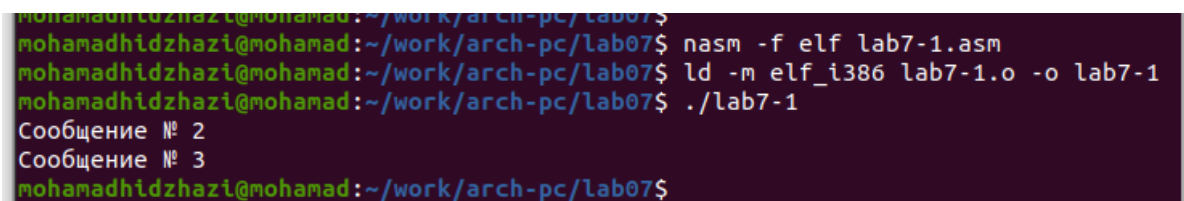


```
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$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

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)



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

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


```
lab7-1.asm — Kate
File Edit View Projects Bookmarks Sessions Tools Settings Help
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
```

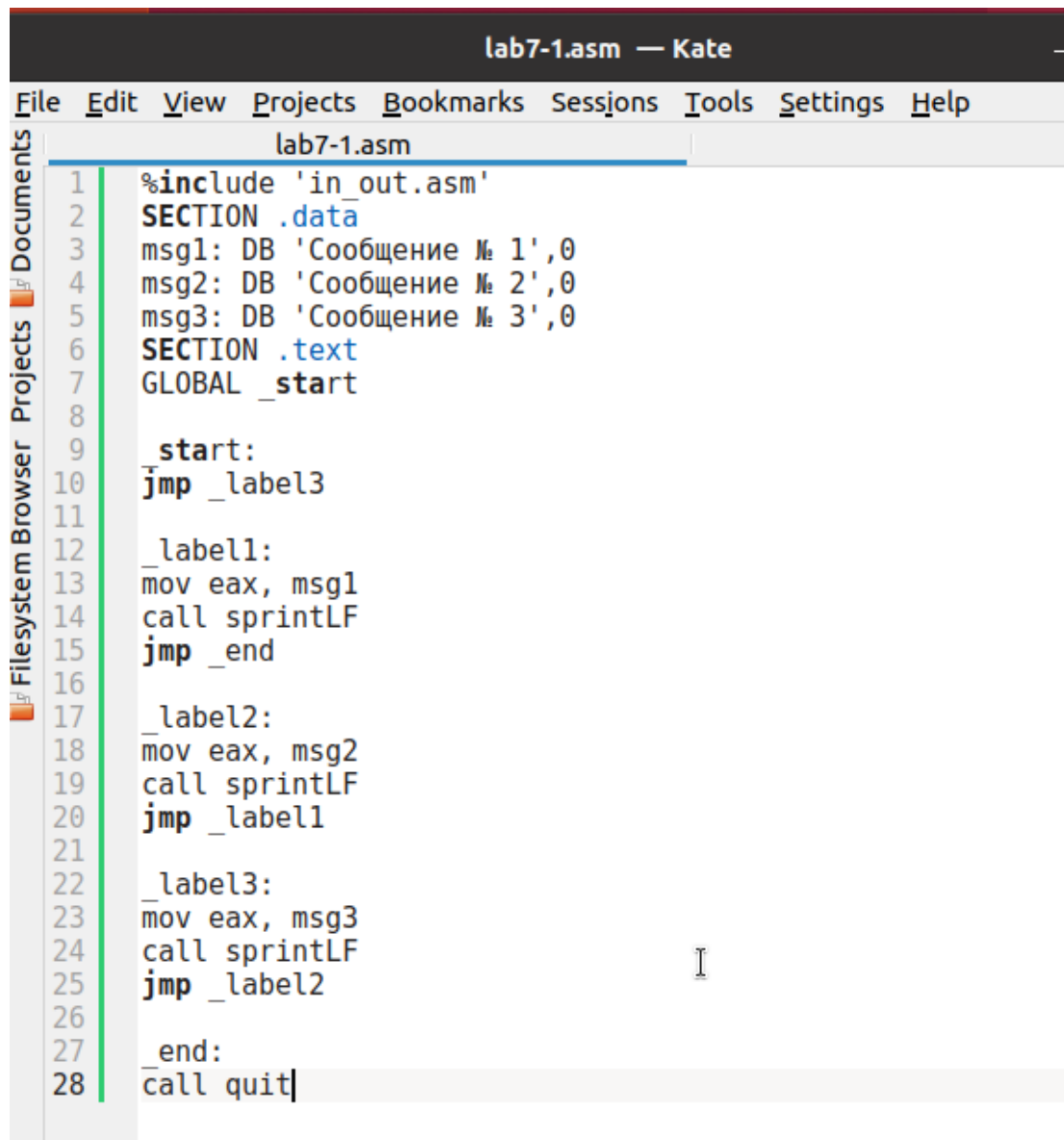
Рис. 2.5: Запуск программы lab7-1.asm

Изменил текст программы, изменив инструкции `jmp`, чтобы вывод программы был следующим (рис. 2.6) (рис. 2.7):

Сообщение № 3

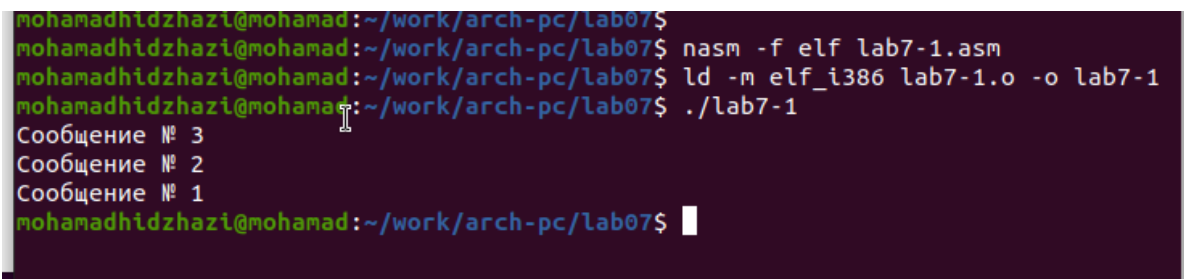
Сообщение № 2

Сообщение № 1



```
lab7-1.asm — Kate
File Edit View Projects Bookmarks Sessions Tools Settings Help
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 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

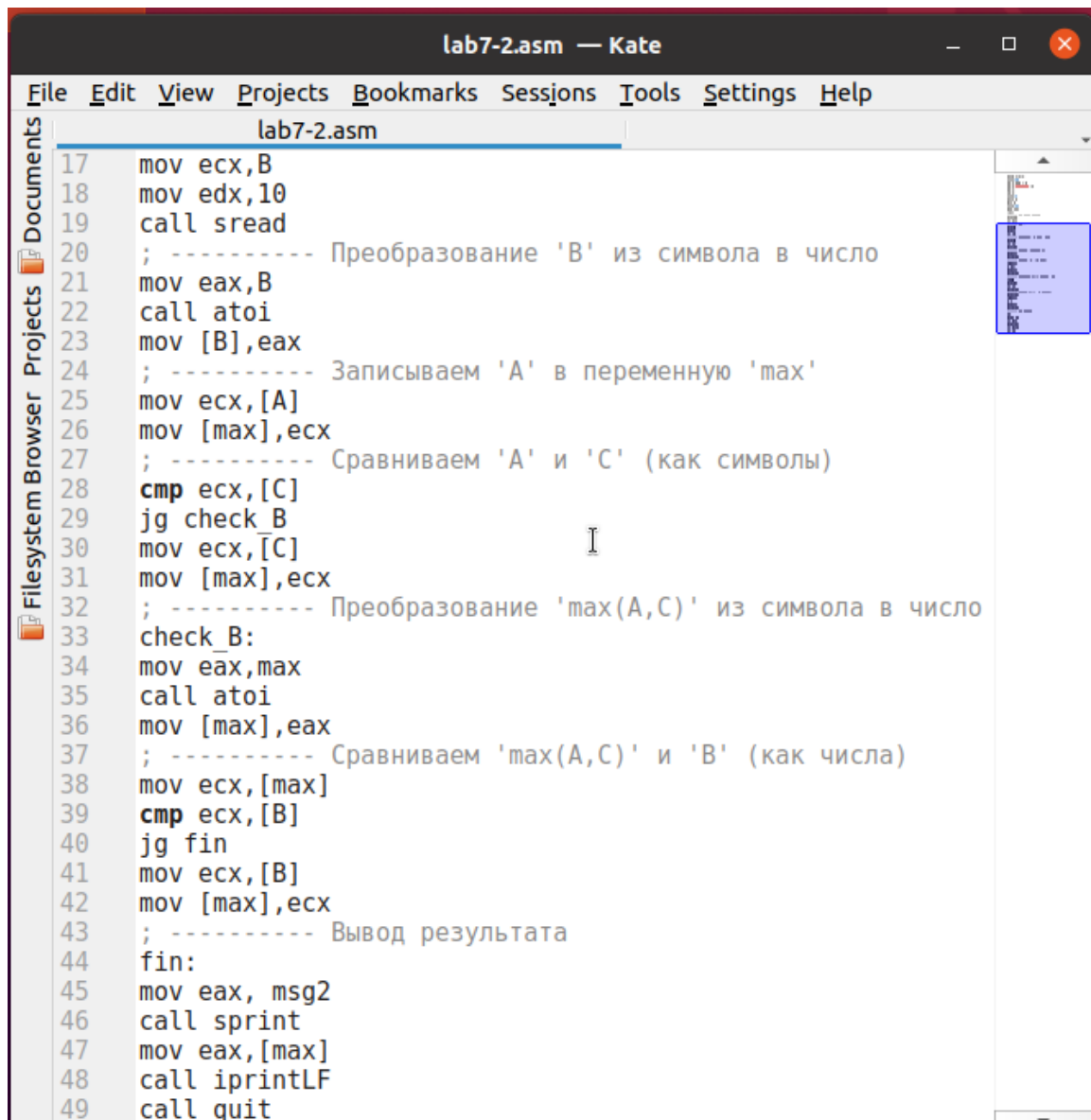


```
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$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

Рис. 2.7: Запуск программы lab7-1.asm

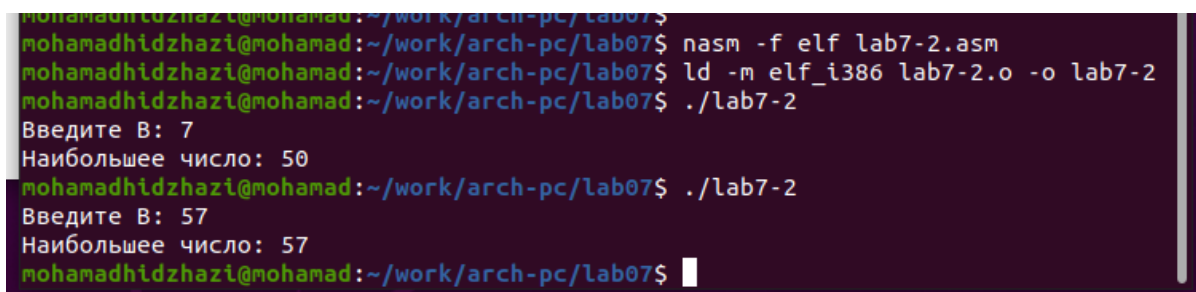
Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую 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
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



```
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 7
Наибольшее число: 50
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 57
Наибольшее число: 57
mohamadhidzhazi@mohamad:~/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
143 142 <1>
144 143 000000BB 80EB30 <1> sub bl, 48
145 144 000000BE 01D8 <1> add eax, ebx
146 145 000000C0 BB0A000000 <1> mov ebx, 10
147 146 000000C5 F7E3 <1> mul ebx
148 147 000000C7 41 <1> inc ecx
149 148 000000C8 EBE2 <1> jmp .multiplyLoop
150 149 <1>
151 150 <1> .finished:
152 151 000000CA 83F900 <1> cmp ecx, 0
153 152 000000CD 7407 <1> je .restore
154 153 000000CF BB0A000000 <1> mov ebx, 10
155 154 000000D4 F7F3 <1> div ebx
156 155 <1>
157 156 <1> .restore:
158 157 000000D6 5E <1> pop esi
159 158 000000D7 5A <1> pop edx
160 159 000000D8 59 <1> pop ecx
161 160 000000D9 5B <1> pop ebx
162 161 000000DA C3 <1> ret
163 162 <1>
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 <1> section .data
173 3 00000000 D092D0B2D0B5D0B4D0- msg1 db 'Введите B: ',0h
174 3 00000009 B8D182D0B520423A20-
175 3 00000012 00

```

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

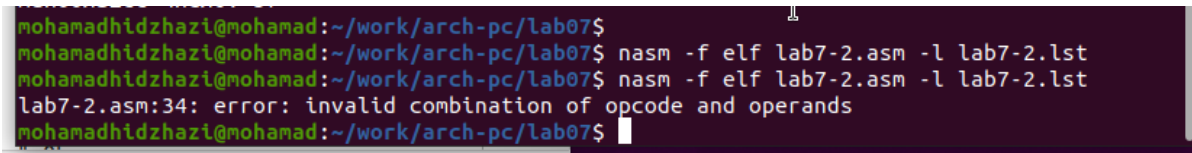
строка 190

- 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)



```
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$  
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
lab7-2.asm:34: error: invalid combination of opcode and operands  
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$
```

Рис. 2.11: Ошибка трансляции lab7-2

```

lab7-2.lst
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 mov eax,
210 34 *****
211 35 00000130 E867FFFFFF error: invalid combination of opcode and operands
212 36 00000135 A3[00000000] call atoi
213 37 mov [max],eax
214 38 0000013A 8B0D[00000000] ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
215 39 00000140 3B0D[0A000000] mov ecx,[max]
216 40 00000146 7F0C cmp ecx,[B]
217 41 00000148 8B0D[0A000000] jg fin
218 42 0000014E 890D[00000000] mov ecx,[B]
219 43 mov [max],ecx
220 44 ; ----- Вывод результата
221 45 00000154 B8[13000000] fin:
222 46 00000159 E8B1FEFFFF mov eax, msg2
223 47 0000015E A1[00000000] call sprint
224 48 00000163 E81EFFFFFF mov eax,[max]
225 49 00000168 E86EFFFFFF call iprintLF
226

```

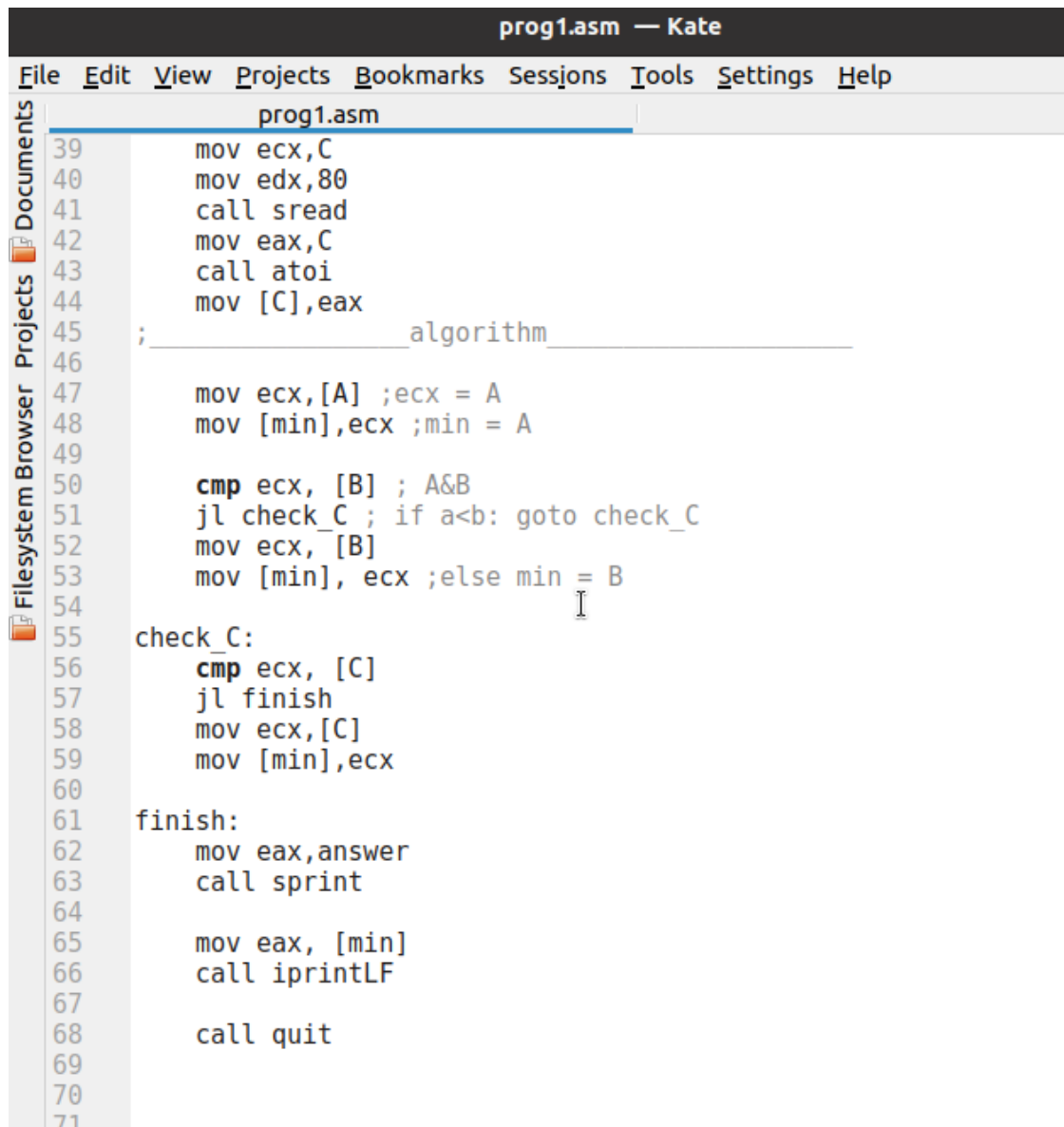
Рис. 2.12: Файл листинга с ошибкой lab7-2

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

2.1 Самостоятельное задание

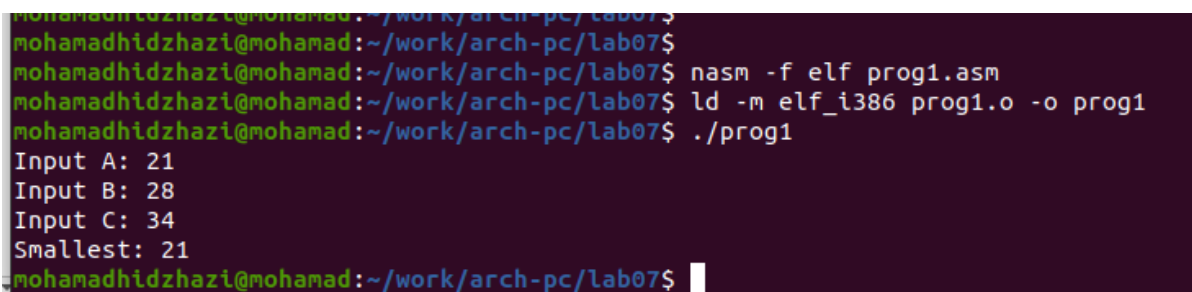
Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу (рис. 2.13) (рис. 2.14)

для варианта 11 - 21,28,34



```
prog1.asm — Kate
File Edit View Projects Bookmarks Sessions Tools Settings Help
prog1.asm
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
71
```

Рис. 2.13: Программа в файле prog1.asm



```
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ nasm -f elf prog1.asm
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ ld -m elf_i386 prog1.o -o prog1
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ ./prog1
Input A: 21
Input B: 28
Input C: 34
Smallest: 21
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$
```

Рис. 2.14: Запуск программы prog1.asm

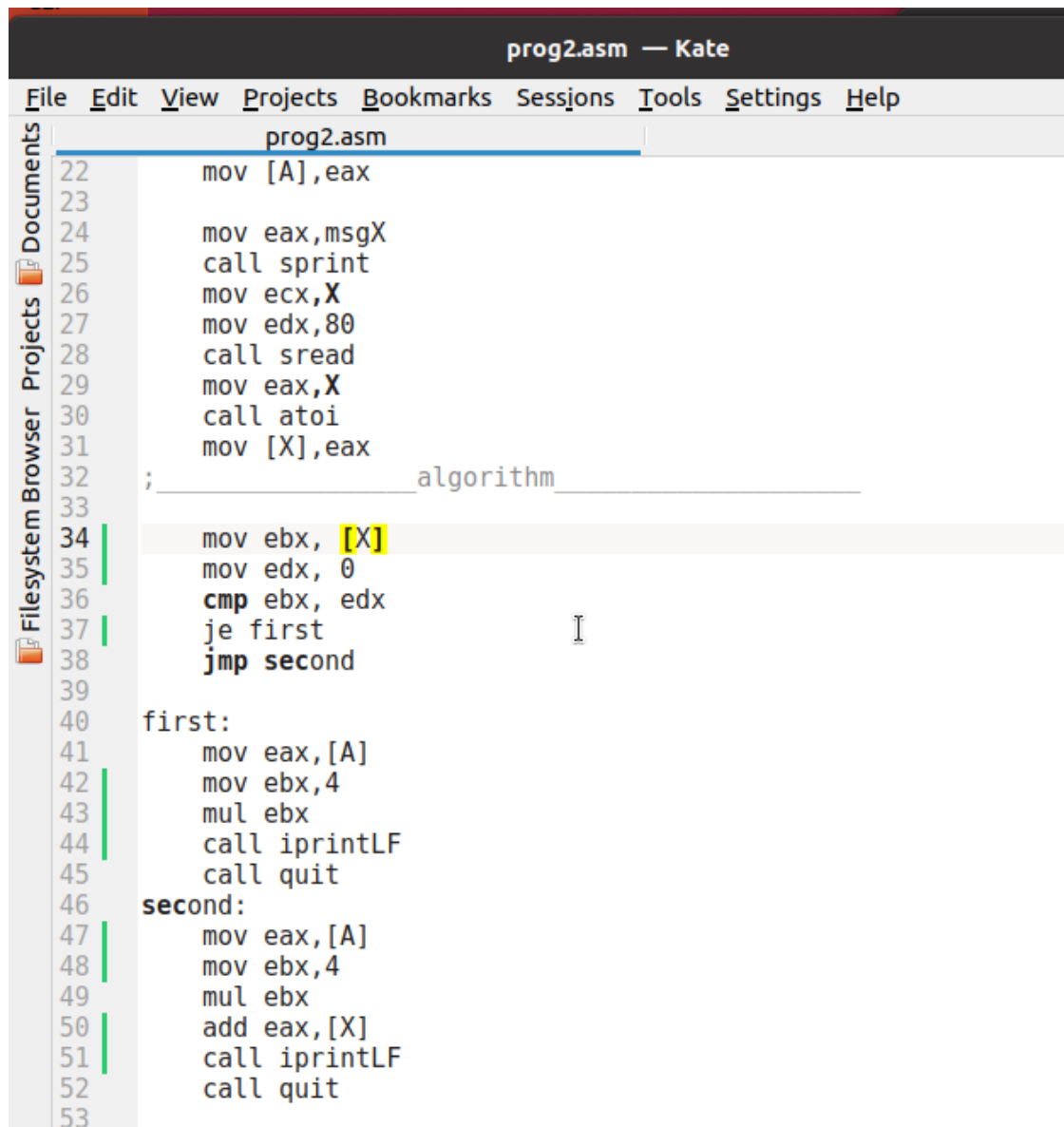
Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений X и a из 7.6. (рис. 2.15) (рис. 2.16)

для варианта 11

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

Если подставить $x = 0, a = 3$ получается 12.

Если подставить $x = 1, a = 2$ получается 9.



```
prog2.asm — Kate
File Edit View Projects Bookmarks Sessions Tools Settings Help
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 ; _____algorithm_____
33
34 mov ebx, [X]
35 mov edx, 0
36 cmp ebx, edx
37 je first
38 jmp second
39
40 first:
41 mov eax,[A]
42 mov ebx,4
43 mul ebx
44 call iprintLF
45 call quit
46 second:
47 mov eax,[A]
48 mov ebx,4
49 mul ebx
50 add eax,[X]
51 call iprintLF
52 call quit
53
```

Рис. 2.15: Программа в файле prog2.asm

```
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$  
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ nasm -f elf prog2.asm  
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ ld -m elf_i386 prog2.o -o prog2  
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ ./prog2  
Input A: 3  
Input X: 0  
12  
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$ ./prog2  
Input A: 2  
Input X: 1  
9  
mohamadhidzhazi@mohamad:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы prog2.asm

3 Выводы

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