

Отчёт по лабораторной работе 7

Архитектура компьютеров и операционные системы

Старикова Владислава Александровна НММбд-03-24

Содержание

0.1	Самостоятельное задание	13
1	Выводы	18

Список иллюстраций

1	Создан каталог	4
2	Программа lab7-1.asm	5
3	Запуск программы lab7-1.asm	6
4	Программа lab7-1.asm	6
5	Запуск программы lab7-1.asm	7
6	Программа в файле lab7-1.asm	8
7	Запуск программы lab7-1.asm	8
8	Программа в файле lab7-2.asm	10
9	Запуск программы lab7-2.asm	10
10	Файл листинга lab7-2	11
11	Ошибка трансляции lab7-2	12
12	Файл листинга с ошибкой lab7-2	13
13	Программа в файле prog1.asm	14
14	Запуск программы prog1.asm	15
15	Программа в файле prog2.asm	16
16	Запуск программы prog2.asm	17

Список таблиц

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

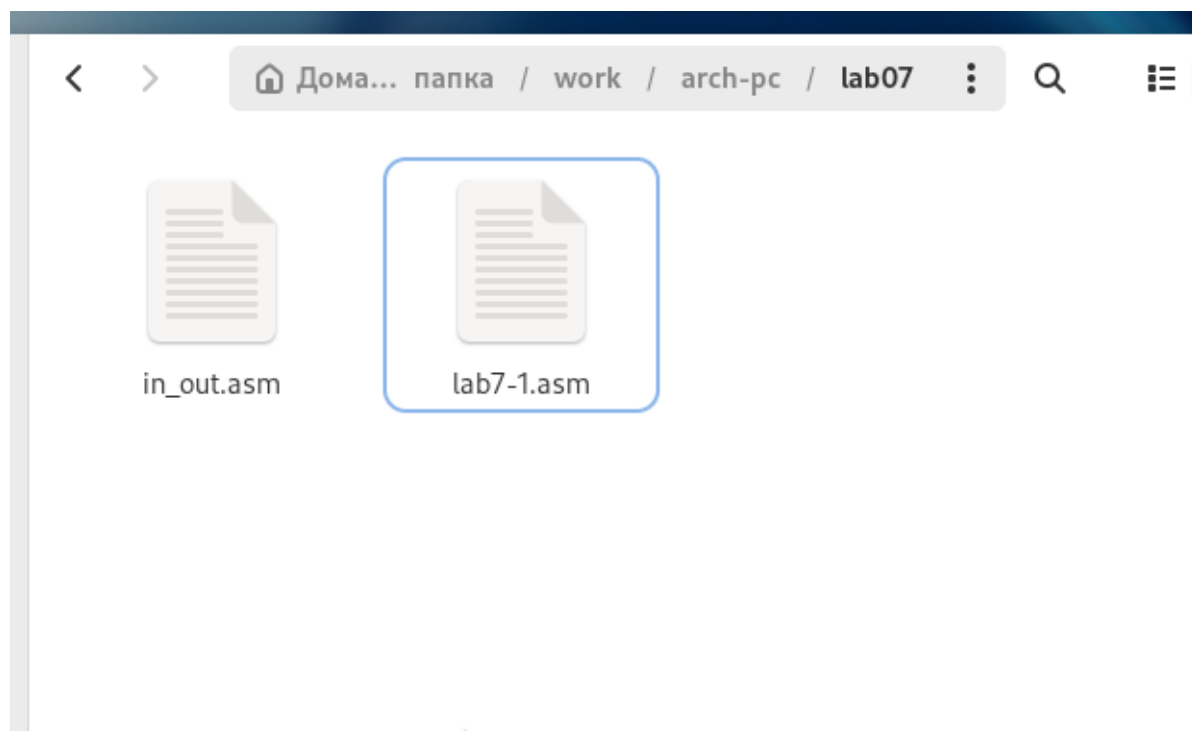
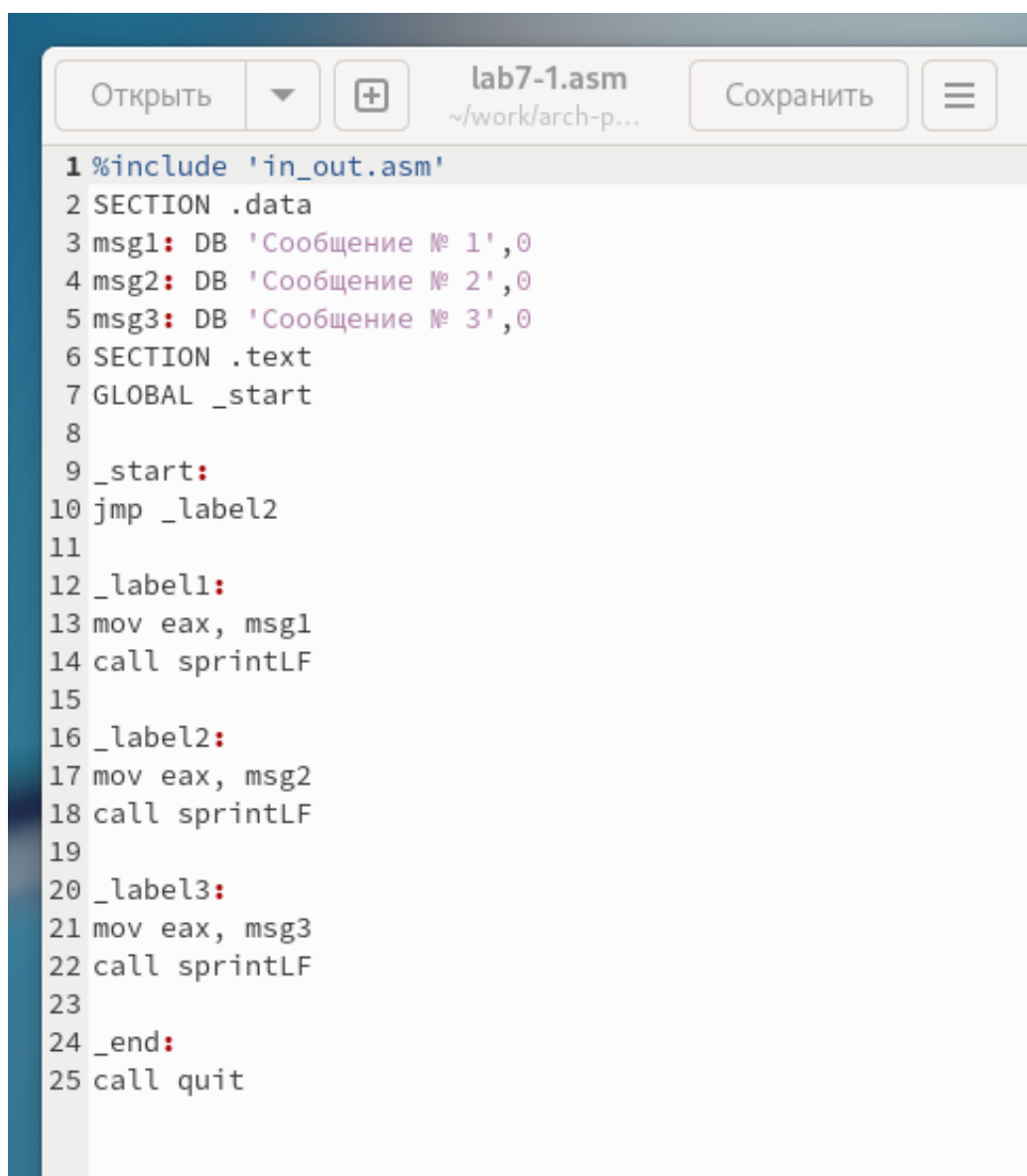


Рис. 1: Создан каталог

В NASM инструкция `jmp` используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. В файле `lab7-1.asm` разместил текст программы из листинга 7.1 (рис. 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
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: Программа lab7-1.asm

Создала исполняемый файл и запустил его (рис. 3).

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

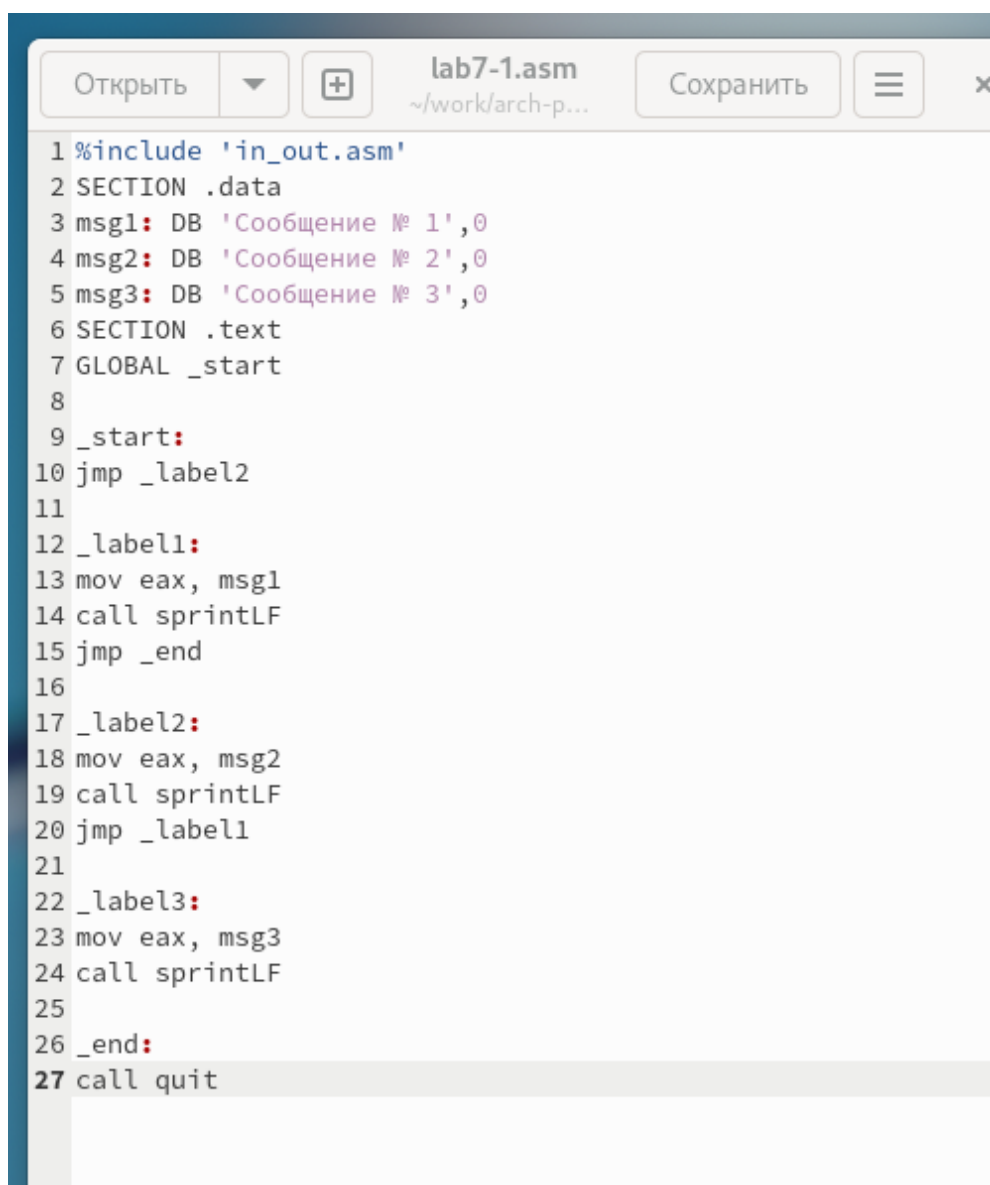
Рис. 3: Запуск программы lab7-1.asm

Инструкция `jmp` позволяет выполнять переходы как вперёд, так и назад. Изменил программу так, чтобы сначала выводилось сообщение № 2, затем сообщение № 1, после чего программа завершала работу. Для этого добавил в текст программы инструкцию `jmp` с меткой `_label1` после вывода сообщения № 2 (чтобы перейти к инструкции вывода сообщения № 1) и инструкцию `jmp` с меткой `_end` после вывода сообщения № 1 (для перехода к инструкции `call quit`).

Обновила текст программы согласно листингу 7.2 (рис. 4 и 5).

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

Рис. 4: Программа lab7-1.asm

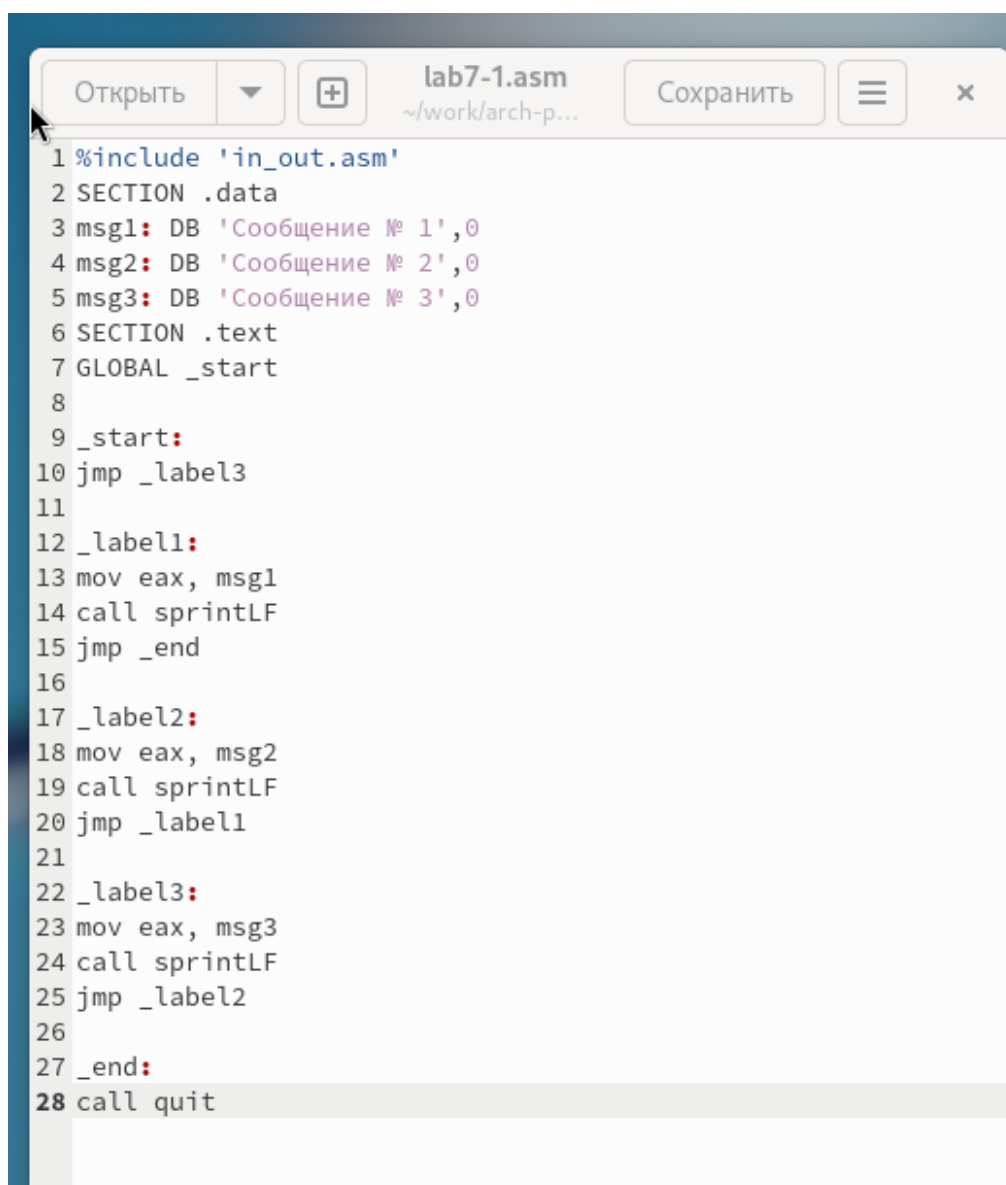


```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8
9 _start:
10 jmp _label2
11
12 _label1:
13 mov eax, msg1
14 call sprintLF
15 jmp _end
16
17 _label2:
18 mov eax, msg2
19 call sprintLF
20 jmp _label1
21
22 _label3:
23 mov eax, msg3
24 call sprintLF
25
26 _end:
27 call quit
```

Рис. 5: Запуск программы lab7-1.asm

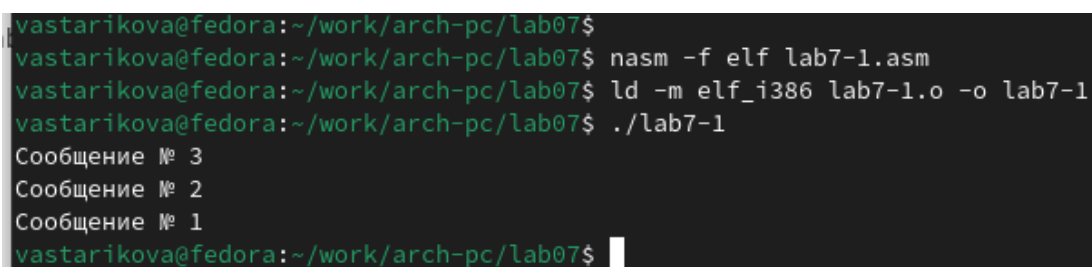
Изменила текст программы так, чтобы итоговый вывод программы выглядел следующим образом (рис. 6 и 7):

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 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. 6: Программа в файле lab7-1.asm

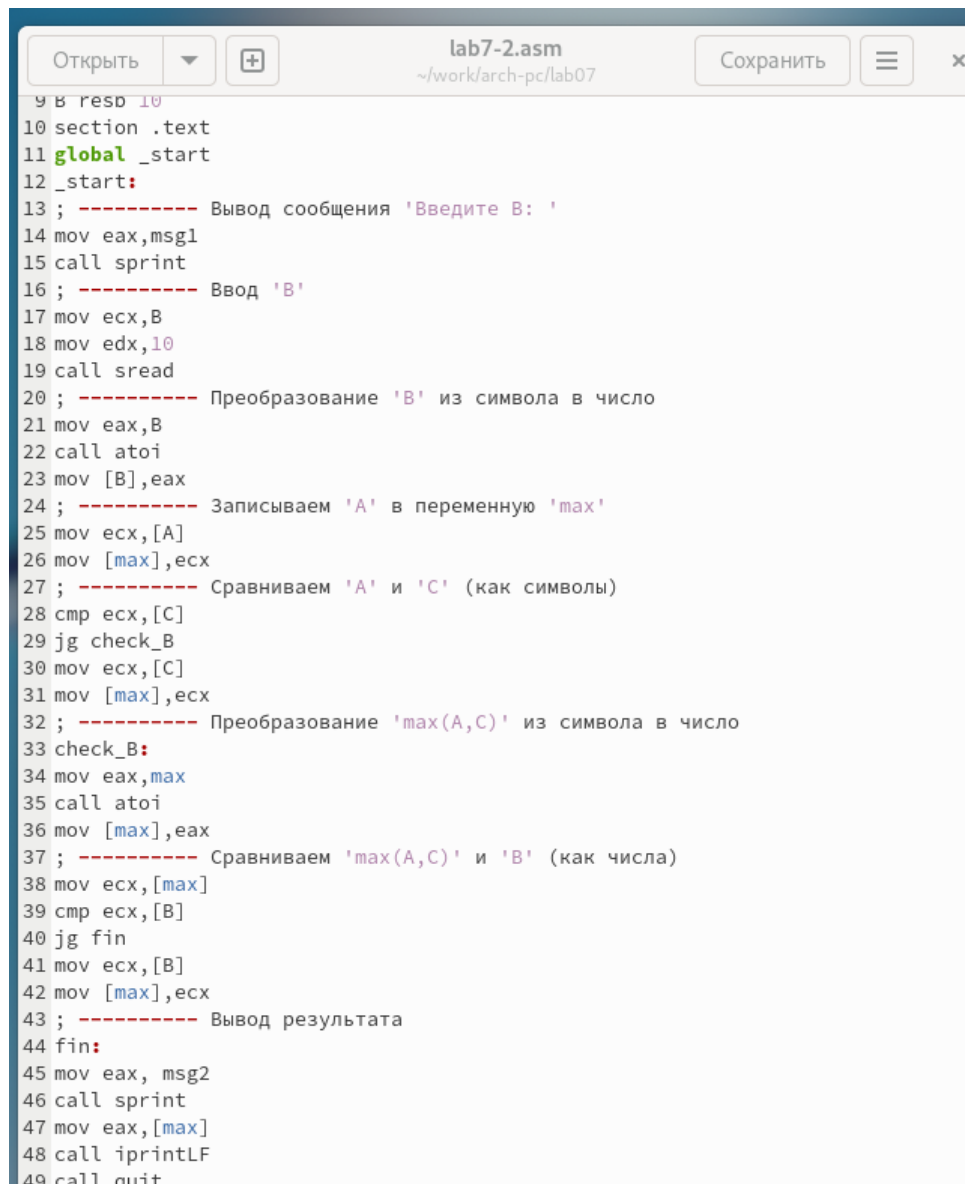


```
vastarikova@fedora:~/work/arch-pc/lab07$
vastarikova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
vastarikova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
vastarikova@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
vastarikova@fedora:~/work/arch-pc/lab07$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

Рис. 7: Запуск программы lab7-1.asm

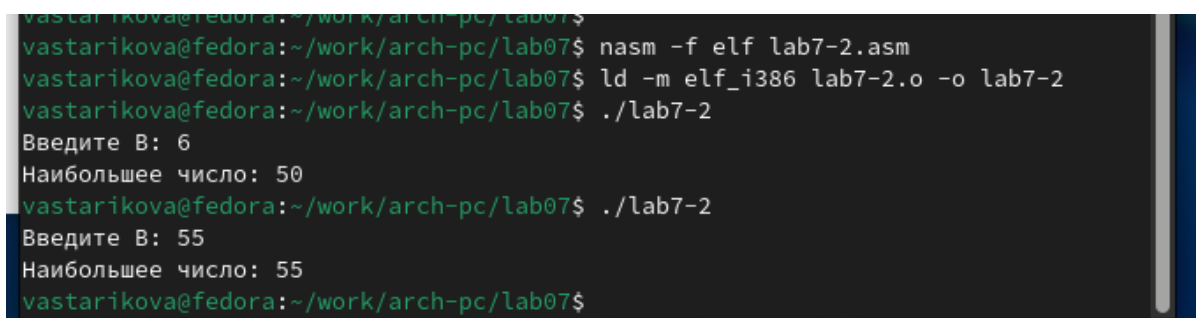
Использование инструкции `jmp` приводит к переходу в любом случае. Однако часто при написании программ требуется использовать условные переходы, когда переход должен происходить при выполнении какого-либо условия. В качестве примера рассмотрим программу, которая находит и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С. Значения для А и С заданы в программе, значение В вводится с клавиатуры.

Я создала исполняемый файл и проверила его работу для разных значений В (рис. 8) (рис. 9).



```
9 B resb 10
10 section .text
11 global _start
12 _start:
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. 8: Программа в файле lab7-2.asm



```
vastarikova@fedora:~/work/arch-pc/lab07$
vastarikova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
vastarikova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
vastarikova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 6
Наибольшее число: 50
vastarikova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 55
Наибольшее число: 55
vastarikova@fedora:~/work/arch-pc/lab07$
```

Рис. 9: Запуск программы lab7-2.asm

Обычно `nasm` создает в результате ассемблирования только объектный файл. Чтобы получить файл листинга, нужно указать ключ `-l` и задать имя файла листинга в командной строке.

Я создала файл листинга для программы из файла `lab7-2.asm` (рис. 10)

```

168 167 000000DB B800000000 <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 Ah> max resb 10
184 9 0000000A <res Ah> 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'
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

```

Рис. 10: Файл листинга lab7-2

Внимательно ознакомилась с его форматом и содержимым. Подробно объяснила содержимое трёх строк файла листинга.

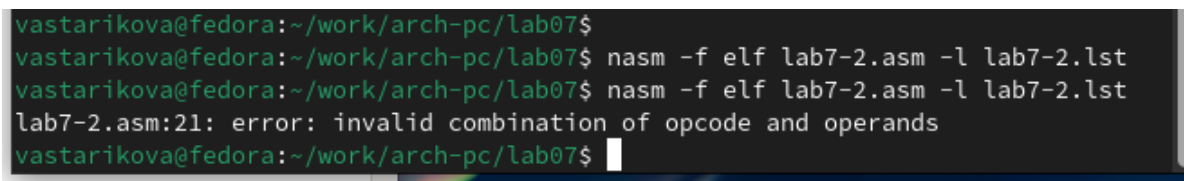
строка 189: - 14 — номер строки в подпрограмме - 000000E8 — адрес - B8[00000000] — машинный код - `mov eax,msg1` — код программы — перекладывает `msg1` в `eax`

строка 190: - 15 — номер строки в подпрограмме - 000000ED — адрес - E81DFFFFFF — машинный код - `call sprint` — код программы — вызов подпрограм-

мы печати

строка 192: - 17 — номер строки в подпрограмме - 000000F2 — адрес - B9[0A000000] — машинный код - mov esx,B — код программы — перекладывает B в esx

Я открыла файл с программой lab7-2.asm и в инструкции с двумя операндами удалила один операнд. Выполнила трансляцию с получением файла листинга. (рис. 11) (рис. 12)

A screenshot of a terminal window with a dark background and green text. The prompt is 'vastarikova@fedora:~/work/arch-pc/lab07\$'. The user enters 'nasm -f elf lab7-2.asm -l lab7-2.lst' twice. The second time, an error message appears: 'lab7-2.asm:21: error: invalid combination of opcode and operands'.

```
vastarikova@fedora:~/work/arch-pc/lab07$  
vastarikova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
vastarikova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
lab7-2.asm:21: error: invalid combination of opcode and operands  
vastarikova@fedora:~/work/arch-pc/lab07$
```

Рис. 11: Ошибка трансляции lab7-2

```

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 Ah>          max resb 10
184      9 0000000A <res Ah>          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                      mov eax,
197     21                      *****
198     22 00000101 E896FFFFFF       error: invalid combination of opcode and operands
199     23 00000106 A3[0A000000]       call atoi
200     24                      mov [B],eax
201     25 0000010B 8B0D[35000000]     ; ----- Записываем 'A' в переменную 'max'
202     26 00000111 890D[00000000]     mov [max],ecx
203     27                      ; ----- Сравниваем 'A' и 'C' (как символы)
204     28 00000117 3B0D[39000000]     cmp ecx,[C]
205     29 0000011D 7F0C              jg check_B
206     30 0000011F 8B0D[39000000]     mov ecx,[C]
207     31 00000125 890D[00000000]     mov [max],ecx
208     32                      ; ----- Преобразование 'max(A,C)' из символа в число

```

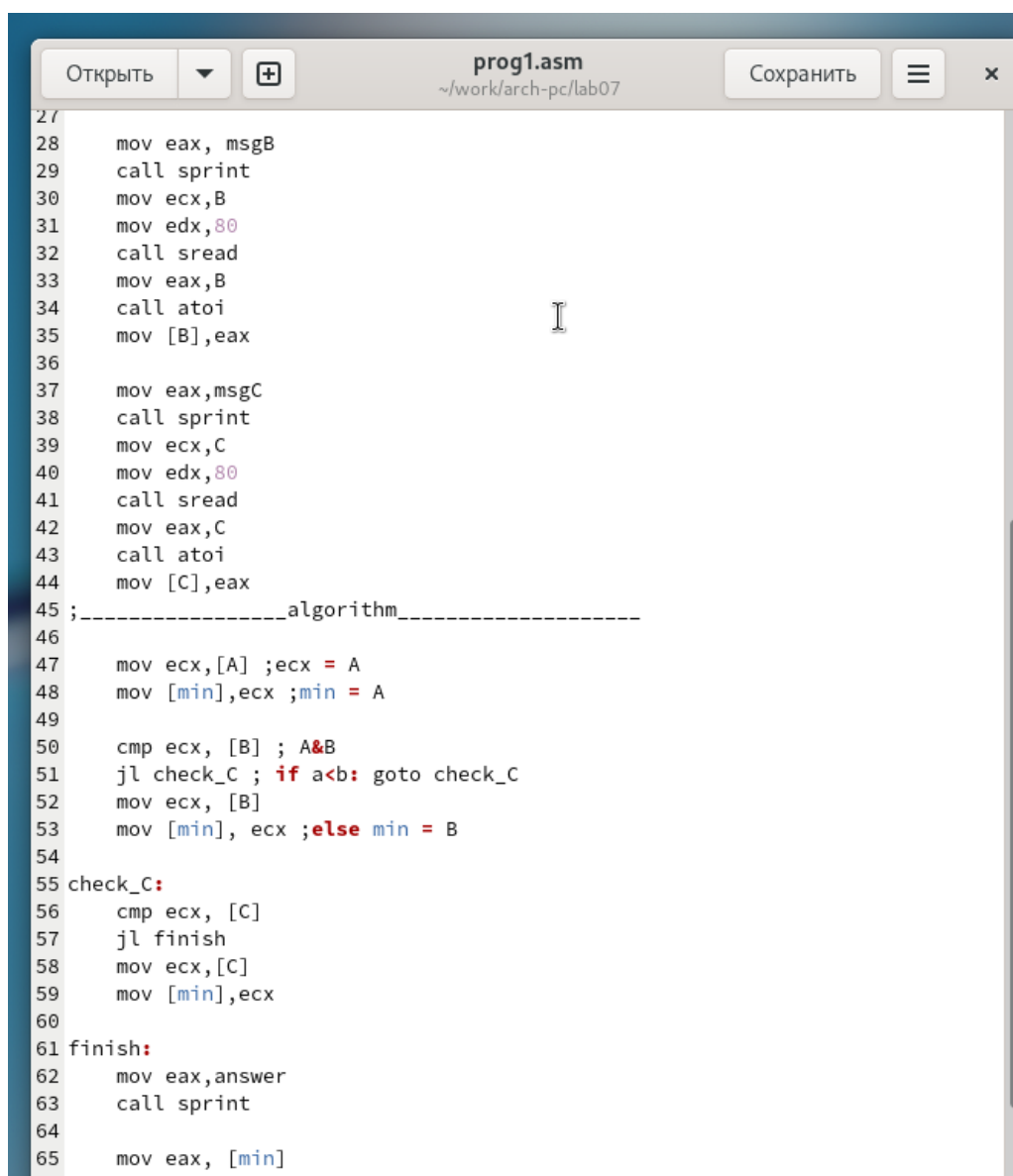
Рис. 12: Файл листинга с ошибкой lab7-2

Объектный файл не смог создаться из-за ошибки. Однако я получила листинг, в котором выделено место ошибки.

0.1 Самостоятельное задание

Напиши 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. Создаю исполняемый файл и проверяю его работу (рис. 13) (рис. 14).

для варианта 17 — 26,12,68



```
27
28     mov eax, msgB
29     call sprint
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 ;-----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 sprint
```

Рис. 13: Программа в файле prog1.asm

```
vastarikova@fedora:~/work/arch-pc/lab07$  
vastarikova@fedora:~/work/arch-pc/lab07$ nasm -f elf prog1.asm  
vastarikova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 prog1.o -o prog1  
vastarikova@fedora:~/work/arch-pc/lab07$ ./prog1  
Input A: 4  
Input B: 3  
Input C: 1  
Smallest: 1  
vastarikova@fedora:~/work/arch-pc/lab07$ ./prog1  
Input A: 26  
Input B: 12  
Input C: 68  
Smallest: 12  
vastarikova@fedora:~/work/arch-pc/lab07$
```

Рис. 14: Запуск программы prog1.asm

Теперь пишу программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбираю из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создаю исполняемый файл и проверяю его работу для значений X и a из 7.6. (рис. 15) (рис. 16).

для варианта 17:

$$\begin{cases} a + 8, a < 8 \\ ax, a \geq 8 \end{cases}$$

Если подставить $x = 3, a = 4$ получается $4 + 8 = 12$.

Если подставить $x = 2, a = 9$ получается $2 * 9 = 18$.

```
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 ;-----algorithm-----
33
34     mov ebx, [A]
35     mov edx, 8
36     cmp ebx, edx
37     jb first
38     jmp second
39
40 first:
41     mov eax,[A]
42     add eax,8
43     call iprintLF
44     call quit
45 second:
46     mov eax,[X]
47     mov ebx,[A]
48     mul ebx
49     call iprintLF
50     call quit
51
52
```

Рис. 15: Программа в файле prog2.asm


```
vastarikova@fedora:~/work/arch-pc/lab07$  
vastarikova@fedora:~/work/arch-pc/lab07$ nasm -f elf prog2.asm  
vastarikova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 prog2.o -o prog2  
vastarikova@fedora:~/work/arch-pc/lab07$ ./prog2  
Input A: 4  
Input X: 3  
12  
vastarikova@fedora:~/work/arch-pc/lab07$ ./prog2  
Input A: 9  
Input X: 2  
18  
vastarikova@fedora:~/work/arch-pc/lab07$
```

Рис. 16: Запуск программы prog2.asm

1 Выводы

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