

Отчёт по лабораторной работе 7

дисциплина: Архитектура компьютера

Кайнова Екатерина Андреевна НПИбд-03-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Условные переходы	11
2.3	Изучение структуры файла листинга	13
2.4	Самостоятельное задание	15
3	Выводы	20

Список иллюстраций

2.1	Создан каталог	6
2.2	Программа lab7-1.asm	7
2.3	Запуск программы lab7-1.asm	7
2.4	Программа lab7-1.asm	8
2.5	Запуск программы lab7-1.asm	9
2.6	Программа lab7-1.asm	10
2.7	Запуск программы lab7-1.asm	10
2.8	Программа 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 task1.asm	16
2.14	Запуск программы task1.asm	16
2.15	Программа task2.asm	18
2.16	Запуск программы task2.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

2.1 Реализация переходов в NASM

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

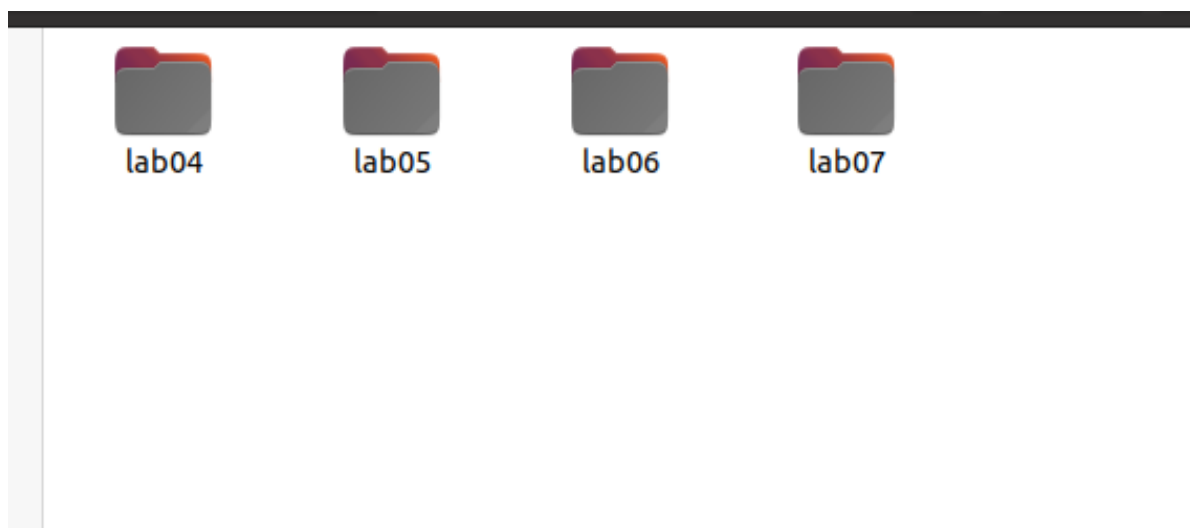
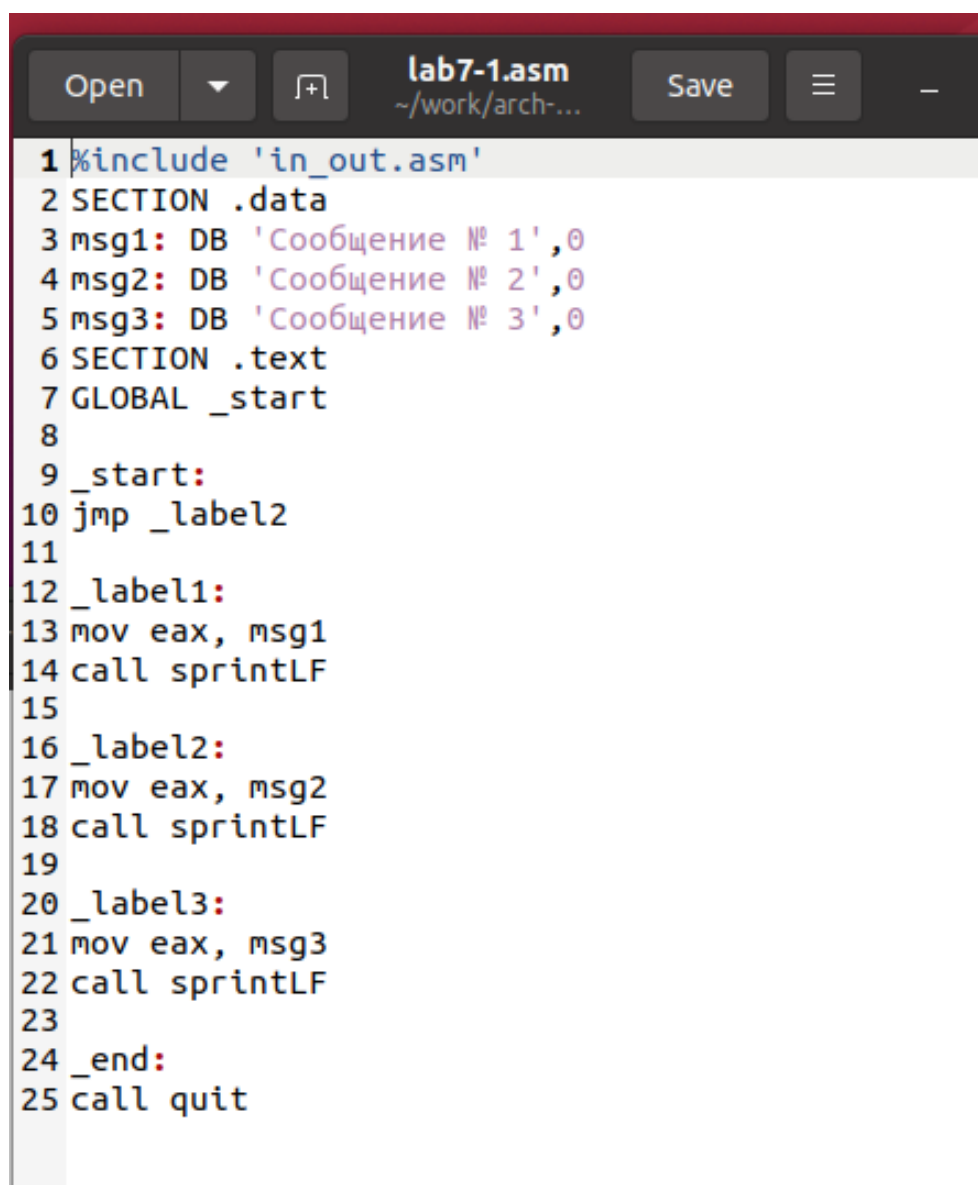


Рис. 2.1: Создан каталог

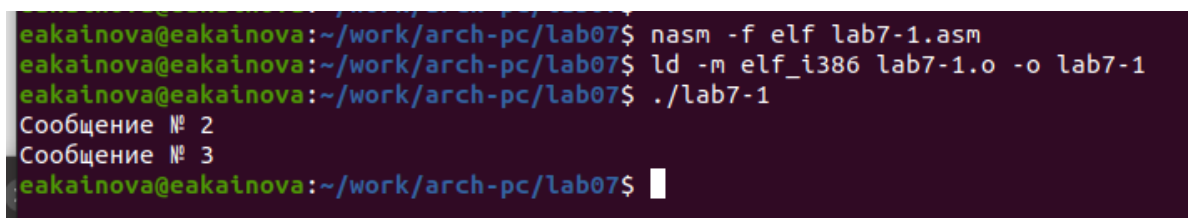
Инструкция `jmp` в NASM используется для реализации безусловных переходов. Пример программы, демонстрирующей эту инструкцию, приведен в файле lab7-1.asm. (рис. 2.2)



```
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 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. 2.2: Программа lab7-1.asm

Создаю исполняемый файл и запускаю его. (рис. 2.3)

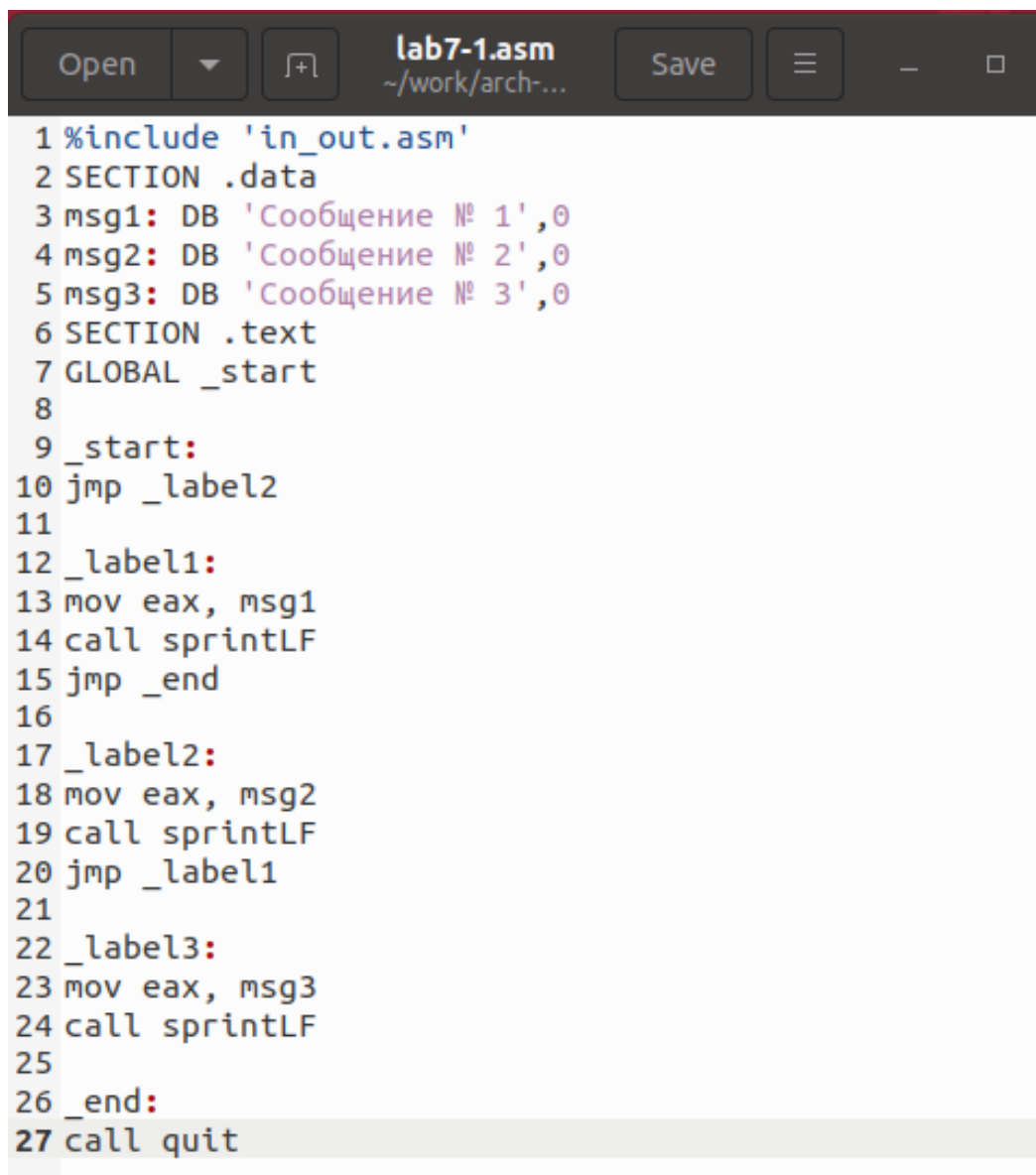


```
eakainova@eakainova:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
eakainova@eakainova:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
eakainova@eakainova:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
eakainova@eakainova:~/work/arch-pc/lab07$
```

Рис. 2.3: Запуск программы lab7-1.asm

Инструкция `jmp` позволяет осуществлять переходы как вперед, так и назад. Для изменения последовательности вывода программы добавляю метки `_label1` и `_end`. Таким образом, вывод программы изменится: сначала отобразится сообщение № 2, затем сообщение № 1, и программа завершит работу.

Обновляю текст программы согласно листингу 7.2. (рис. 2.4, рис. 2.5)



```
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. 2.4: Программа lab7-1.asm


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

Рис. 2.5: Запуск программы lab7-1.asm

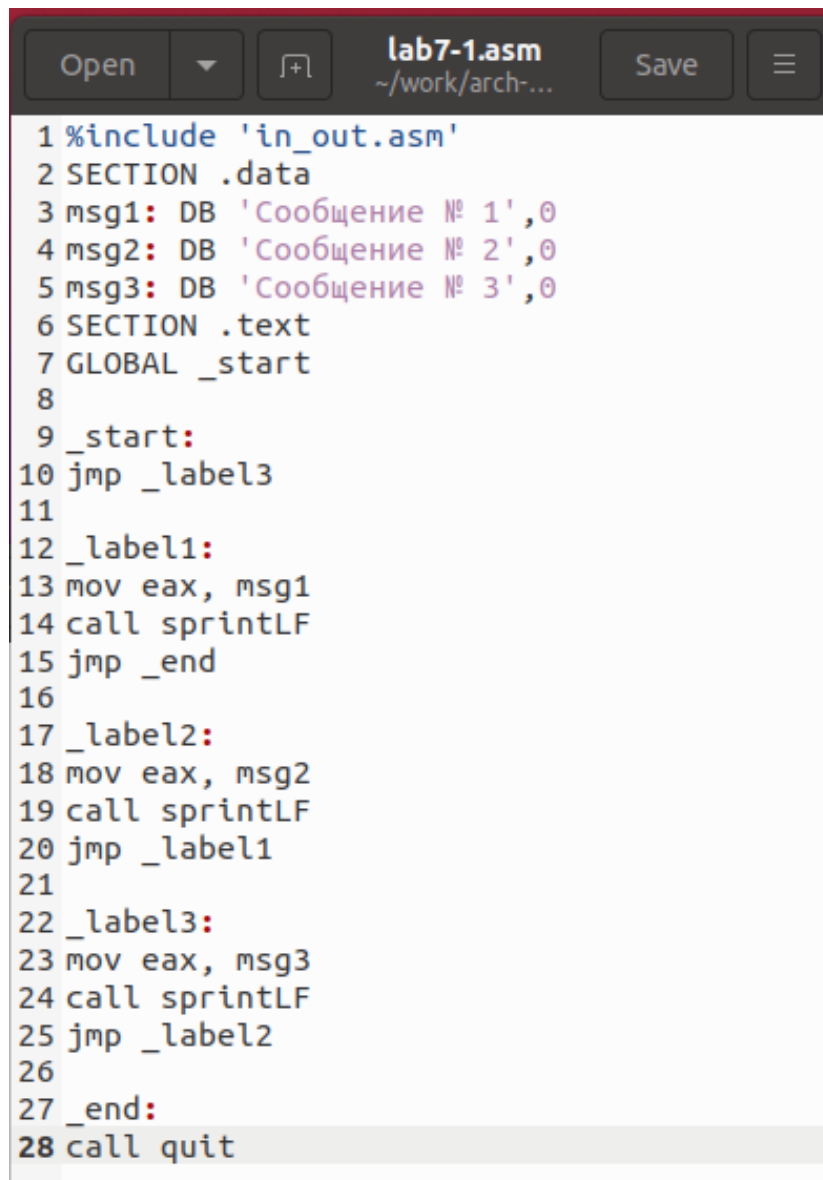
Дорабатываю текст программы для вывода следующих сообщений:

Сообщение № 3

Сообщение № 2

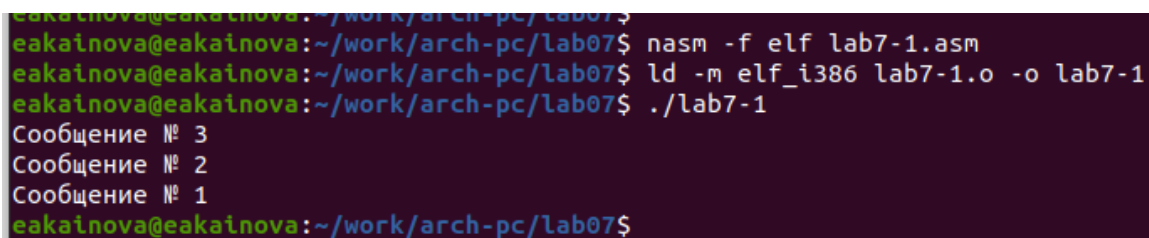
Сообщение № 1

Результат показан на рисунках (рис. 2.6, рис. 2.7).



```
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. 2.6: Программа lab7-1.asm



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

Рис. 2.7: Запуск программы lab7-1.asm

Использование инструкции `jmp` обеспечивает переходы независимо от условий. Однако для реализации условных переходов требуется использование дополнительных инструкций.

2.2 Условные переходы

Для демонстрации условных переходов создаю программу, определяющую максимальное значение среди трех переменных: А, В и С. Значения А и С задаются в программе, а В вводится с клавиатуры. Результаты работы программы представлены на рисунках (рис. 2.8, рис. 2.9).

```

lab7-2.asm
~/work/arch-pc/lab07
Open Save
1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите В: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '20'
6 C dd '50'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите В: '
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод 'В'
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'В' из символа в число
21 mov eax,B
22 call atoi
23 mov [B],eax
24 ; ----- Записываем 'А' в переменную 'max'
25 mov ecx,[A]
26 mov [max],ecx
27 ; ----- Сравниваем 'А' и 'С' (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 'В' (как числа)
38 mov ecx,[max]

```

Рис. 2.8: Программа lab7-2.asm

```

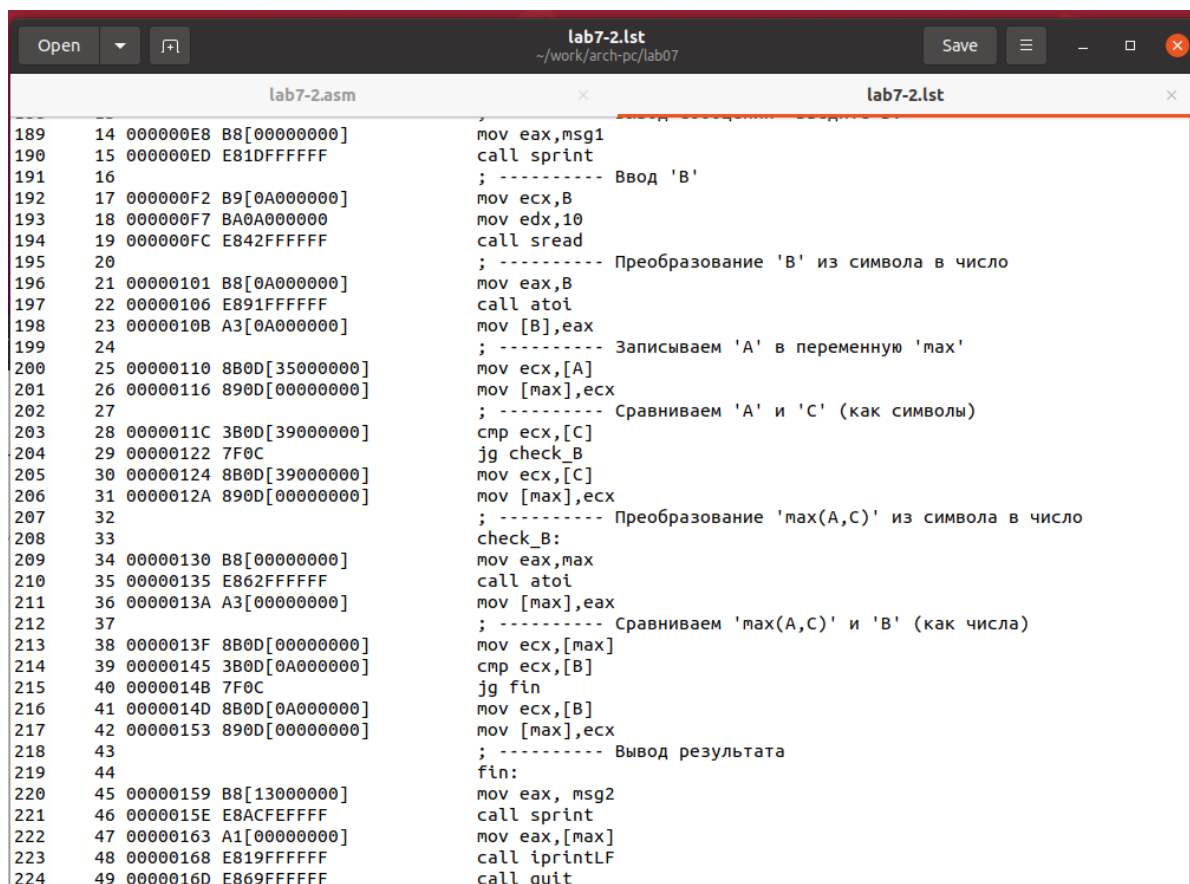
eakainova@eakainova:~/work/arch-pc/lab07$
eakainova@eakainova:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
eakainova@eakainova:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
eakainova@eakainova:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 30
Наибольшее число: 50
eakainova@eakainova:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 60
Наибольшее число: 60
eakainova@eakainova:~/work/arch-pc/lab07$

```

Рис. 2.9: Запуск программы lab7-2.asm

2.3 Изучение структуры файла листинга

Для получения файла листинга указываю ключ `-l` при ассемблировании. Результат ассемблирования программы `lab7-2.asm` представлен на рисунке (рис. 2.10).



```
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]
214 39 00000145 3B0D[0A000000]    cmp ecx,[B]
215 40 0000014B 7F0C                                jg fin
216 41 0000014D 8B0D[0A000000]    mov ecx,[B]
217 42 00000153 890D[00000000]    mov [max],ecx
218 43                                ; ----- Вывод результата
219 44                                fin:
220 45 00000159 B8[13000000]    mov eax, msg2
221 46 0000015E E8ACFFFFFF    call sprint
222 47 00000163 A1[00000000]    mov eax,[max]
223 48 00000168 E819FFFFFF    call iprintLF
224 49 0000016D E869FFFFFF    call quit
```

Рис. 2.10: Файл листинга lab7-2

Анализируя структуру листинга, можно увидеть соответствие строк кода и их машинного представления. Например:

- **Строка 203:**
 - Номер строки: 28
 - Адрес: 0000011C

- Машинный код: 3B0D[39000000]
- Команда: `str esx,[C]`

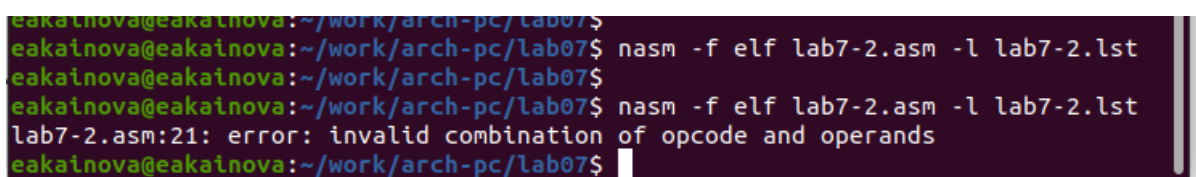
- **Строка 204:**

- Номер строки: 29
- Адрес: 00000122
- Машинный код: 7F0C
- Команда: `jb check_B`

- **Строка 205:**

- Номер строки: 30
- Адрес: 00000124
- Машинный код: 8B0D[39000000]
- Команда: `mov esx,[C]`

Далее изменяю инструкцию с двумя операндами, удаляя один, и повторяю трансляцию. Возникает ошибка, результат которой отображен на рисунках (рис. 2.11, рис. 2.12).



```
eakainova@eakainova:~/work/arch-pc/lab07$
eakainova@eakainova:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
eakainova@eakainova:~/work/arch-pc/lab07$
eakainova@eakainova:~/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
eakainova@eakainova:~/work/arch-pc/lab07$
```

Рис. 2.11: Ошибка трансляции lab7-2

```

lab7-2.lst
~/work/arch-pc/lab07

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 00000108 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
209 33 check_B:
210 34 0000012B B8[00000000] mov eax,max
211 35 00000130 E867FFFFFF call atoi
212 36 00000135 A3[00000000] mov [max],eax
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 E8B1FFFFFF call sprint
223 47 0000015E A1[00000000] mov eax,[max]
224 48 00000163 E81EFFFFFF call iprintLF

```

Рис. 2.12: Файл листинга с ошибкой lab7-2

2.4 Самостоятельное задание

1. Напишите программу, которая находит наименьшее значение из трех переменных a, b и c для следующих значений:

Вариант 11: 21,28,34.

Результат работы программы показан на рисунках (рис. 2.13, рис. 2.14).

```

Open ▼ [+]
```

```

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
68
69

```

Рис. 2.13: Программа task1.asm

```

eakainova@eakainova:~/work/arch-pc/lab07$ nasm -f elf task1.asm
eakainova@eakainova:~/work/arch-pc/lab07$ ld -m elf_i386 task1.o -o task1
eakainova@eakainova:~/work/arch-pc/lab07$ ./task1
Input A: 21
Input B: 28
Input C: 34
Smallest: 21
eakainova@eakainova:~/work/arch-pc/lab07$

```

Рис. 2.14: Запуск программы task1.asm

2. Напишите программу для вычисления функции $f(x)$ для введенных значений x и a :

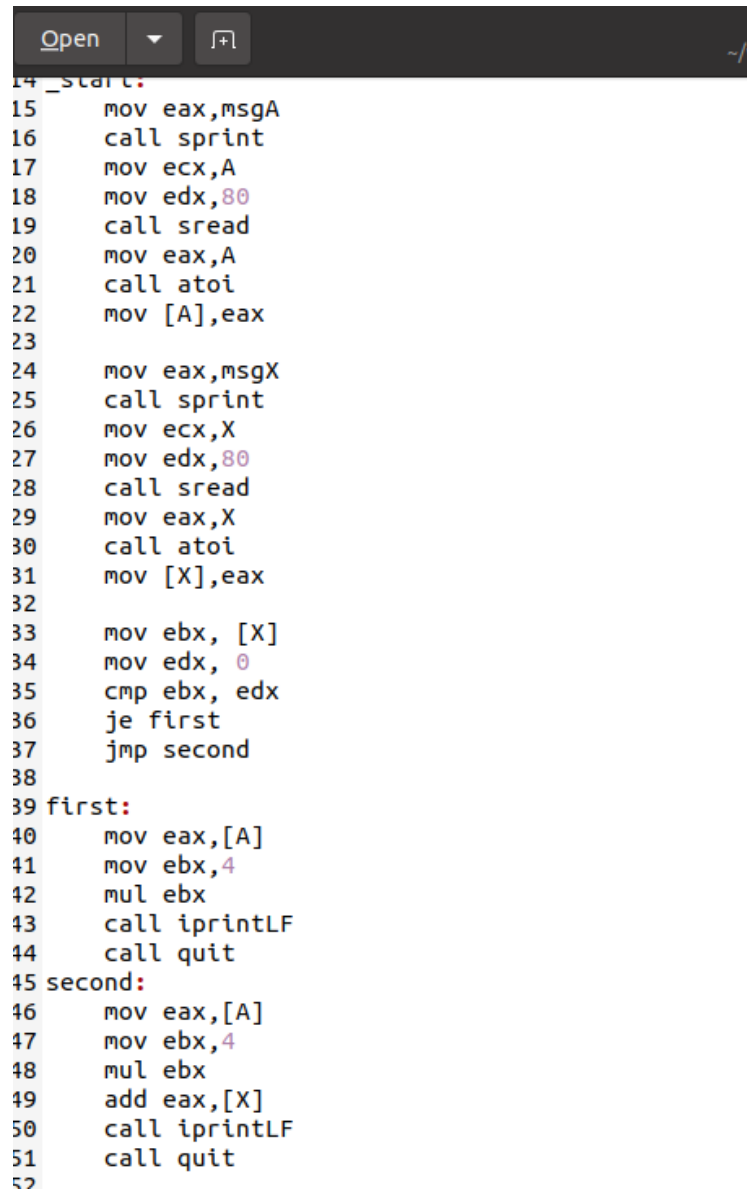
Вариант 11:

$$f(x) = \begin{cases} 4a, & \text{если } x = 0 \\ 4a + x, & \text{если } x \neq 0 \end{cases}$$

При $x = 0, a = 3$ результат: 12.

При $x = 1, a = 2$ результат: 9.

Результаты программы представлены на рисунках (рис. 2.15, рис. 2.16).



```

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 ebx, [X]
34     mov edx, 0
35     cmp ebx, edx
36     je first
37     jmp second
38
39 first:
40     mov eax,[A]
41     mov ebx,4
42     mul ebx
43     call iprintLF
44     call quit
45 second:
46     mov eax,[A]
47     mov ebx,4
48     mul ebx
49     add eax,[X]
50     call iprintLF
51     call quit
52

```

Рис. 2.15: Программа task2.asm

```
eakainova@eakainova:~/work/arch-pc/lab07$  
eakainova@eakainova:~/work/arch-pc/lab07$ nasm -f elf task2.asm  
eakainova@eakainova:~/work/arch-pc/lab07$ ld -m elf_i386 task2.o -o task2  
eakainova@eakainova:~/work/arch-pc/lab07$ ./task2  
Input A: 3  
Input X: 0  
12  
eakainova@eakainova:~/work/arch-pc/lab07$ ./task2  
Input A: 2  
Input X: 1  
9  
eakainova@eakainova:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы task2.asm

3 Выводы

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