

Отчёт по лабораторной работе 7

Архитектура компьютеров

Сюй Хайфэн

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файла листинга	14
2.3	Самостоятельное задание	16
3	Выводы	21

Список иллюстраций

2.1	Создан каталог	7
2.2	Программа lab7-1.asm	8
2.3	Запуск программы lab7-1.asm	8
2.4	Программа lab7-1.asm	9
2.5	Запуск программы lab7-1.asm	10
2.6	Программа lab7-1.asm	11
2.7	Запуск программы lab7-1.asm	11
2.8	Программа lab7-2.asm	13
2.9	Запуск программы lab7-2.asm	13
2.10	Файл листинга lab7-2	14
2.11	Ошибка трансляции lab7-2	15
2.12	Файл листинга с ошибкой lab7-2	16
2.13	Программа lab7-task1.asm	17
2.14	Запуск программы lab7-task1.asm	17
2.15	Программа lab7-task2.asm	19
2.16	Запуск программы lab7-task2.asm	20

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)

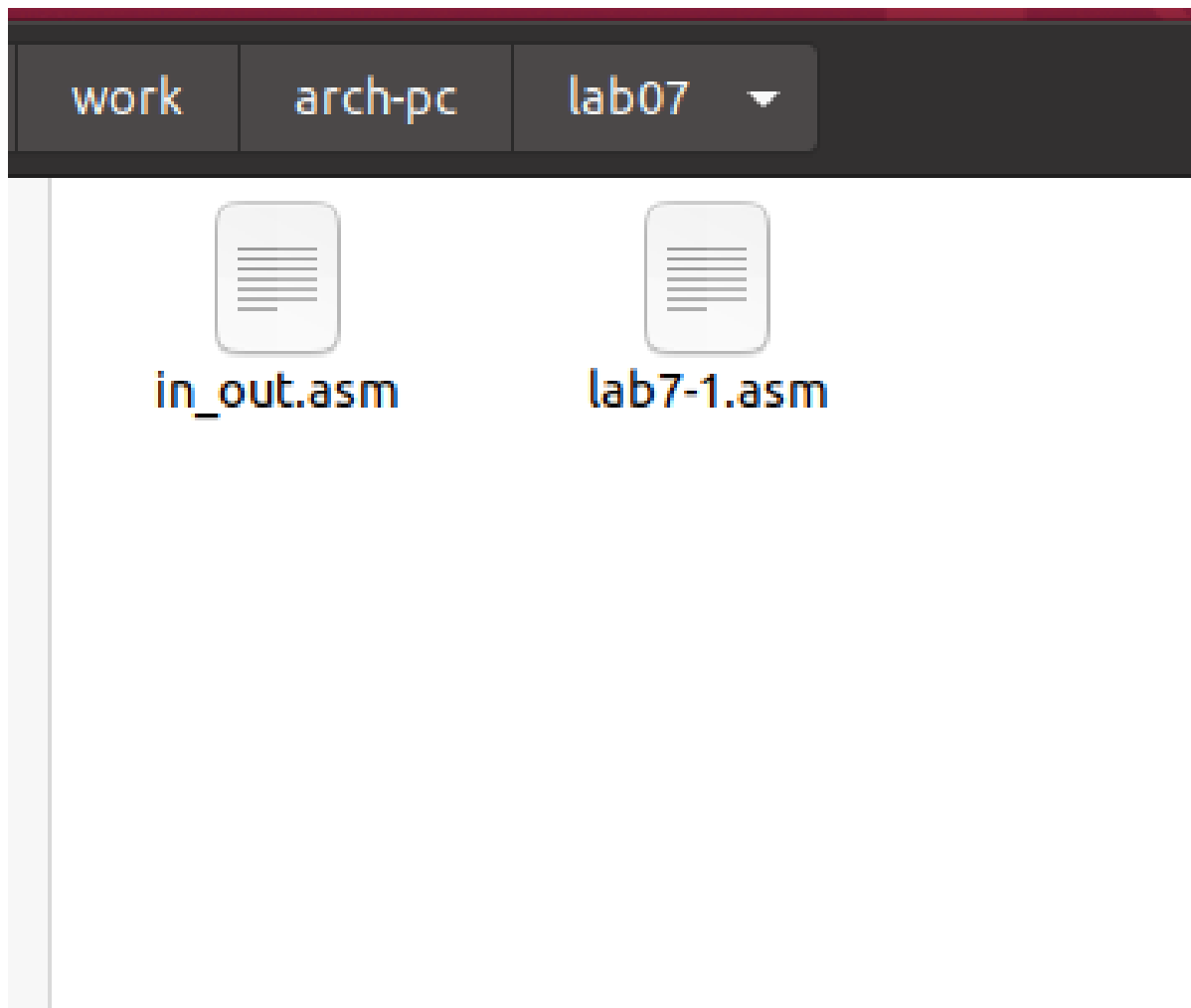
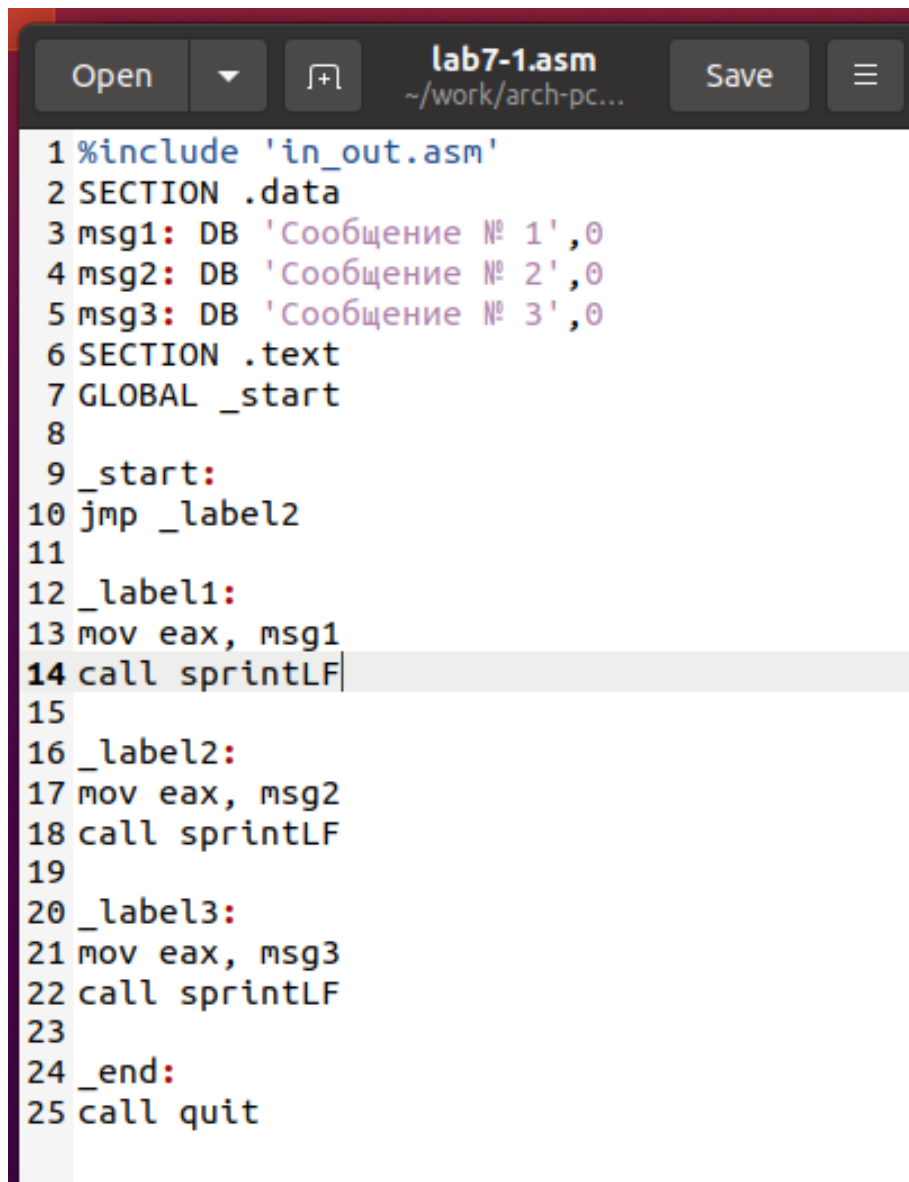


Рис. 2.1: Создан каталог

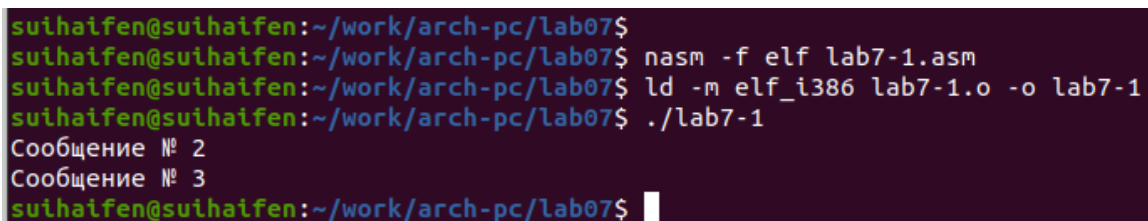
Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Написал в файл `lab7-1.asm` текст программы из листинга 7.1. (рис. 2.2)



```
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 printf
15
16 _label2:
17 mov eax, msg2
18 call printf
19
20 _label3:
21 mov eax, msg3
22 call printf
23
24 _end:
25 call quit
```

Рис. 2.2: Программа lab7-1.asm

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

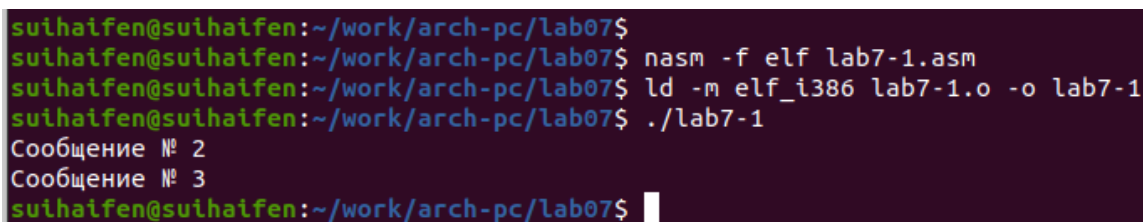


```
suihaifen@suihaifen:~/work/arch-pc/lab07$
suihaifen@suihaifen:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
suihaifen@suihaifen:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
suihaifen@suihaifen:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
suihaifen@suihaifen:~/work/arch-pc/lab07$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

Рис. 2.3: Запуск программы lab7-1.asm

Инструкция `jmp` позволяет осуществлять переходы не только вперед, но и назад. Изменим программу так, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавляем инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавляем инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`).

Изменяю текст программы в соответствии с листингом 7.2. (рис. 2.4) (рис. 2.5)



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

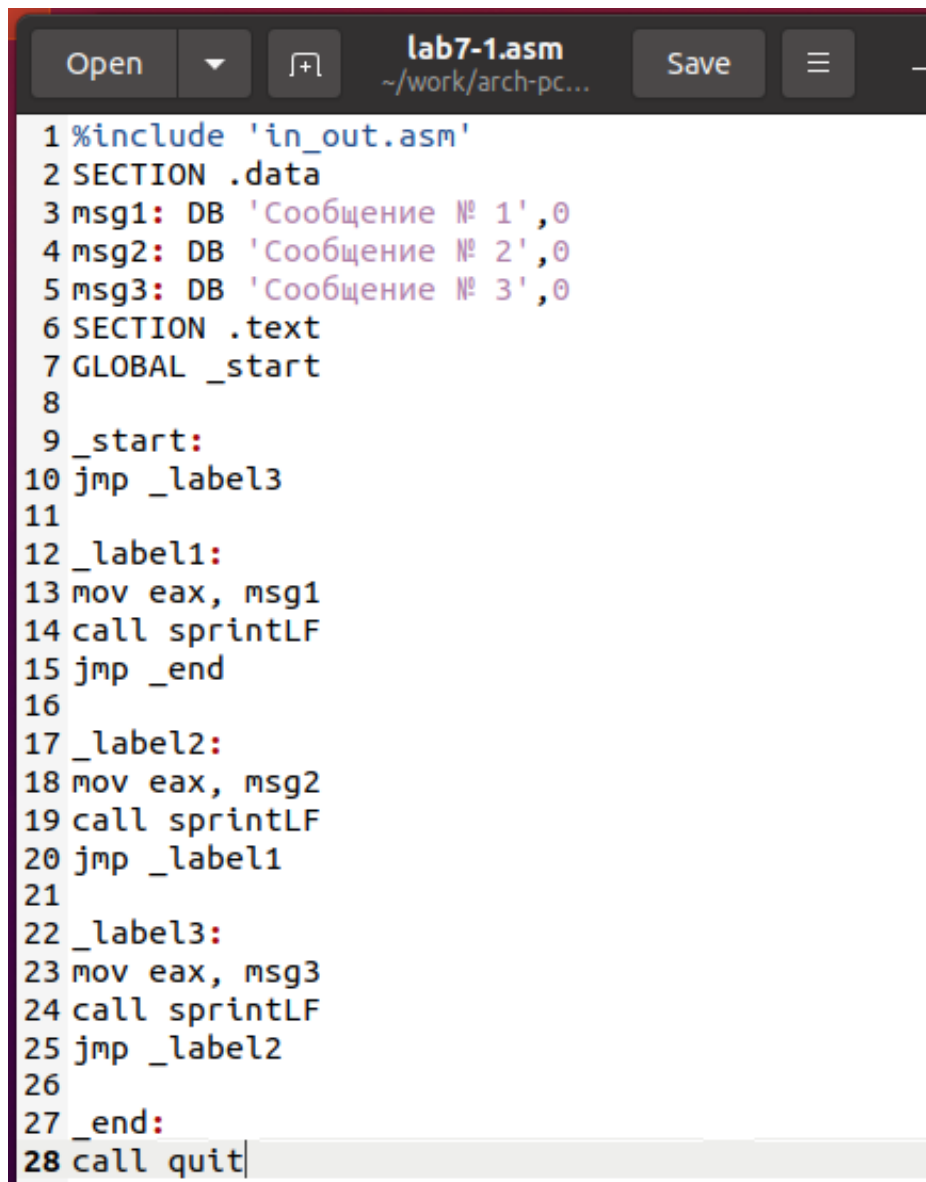
Рис. 2.4: Программа lab7-1.asm

```
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 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. 2.5: Запуск программы lab7-1.asm

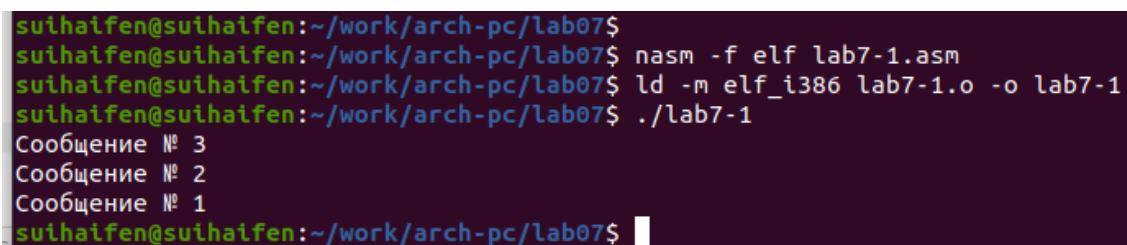
Изменил текст программы, чтобы вывод программы был следующим (рис. 2.6) (рис. 2.7):

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 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 lab7-1.asm

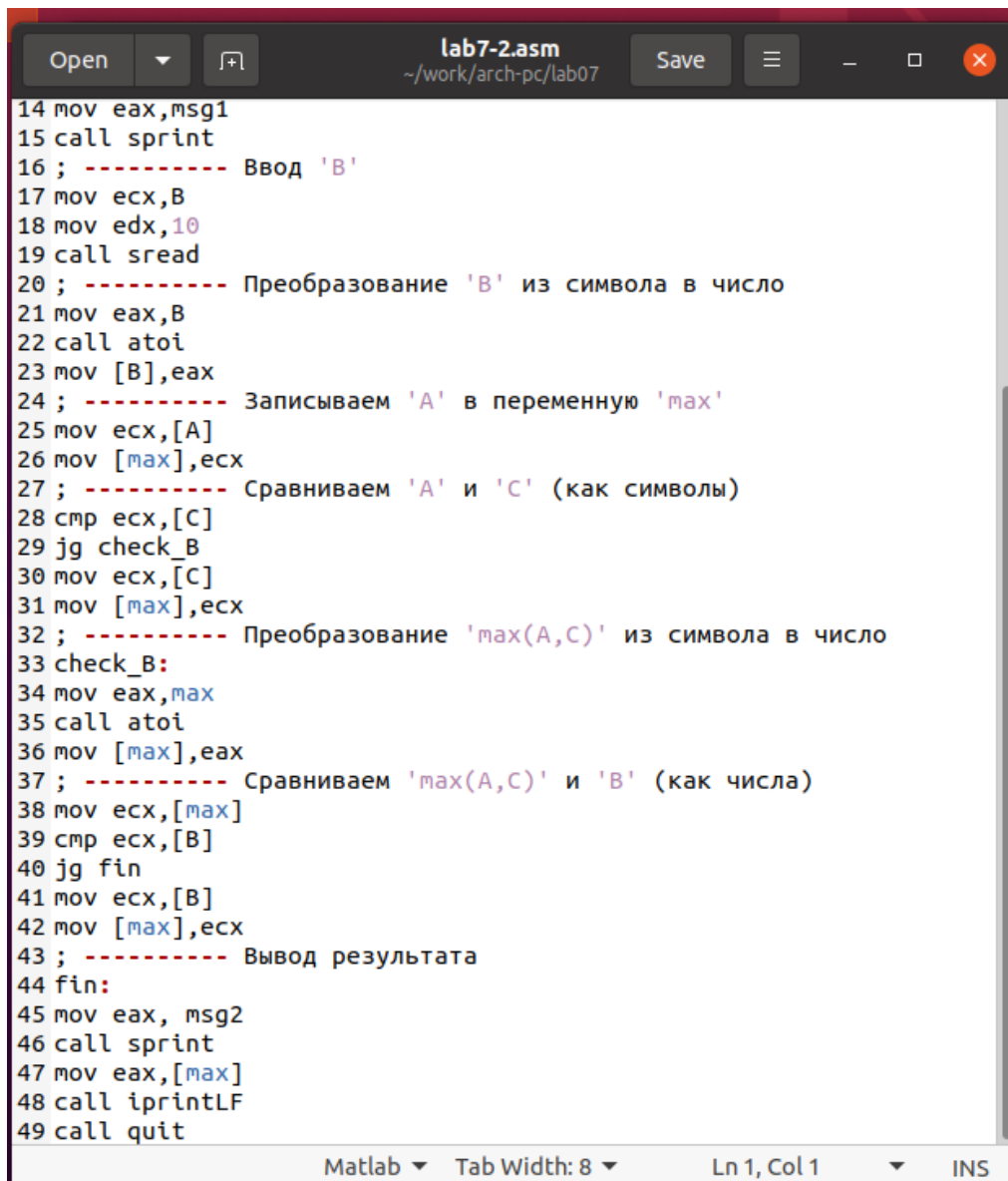


```
suihaifen@suihaifen:~/work/arch-pc/lab07$
suihaifen@suihaifen:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
suihaifen@suihaifen:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
suihaifen@suihaifen:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
suihaifen@suihaifen:~/work/arch-pc/lab07$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

Рис. 2.7: Запуск программы lab7-1.asm

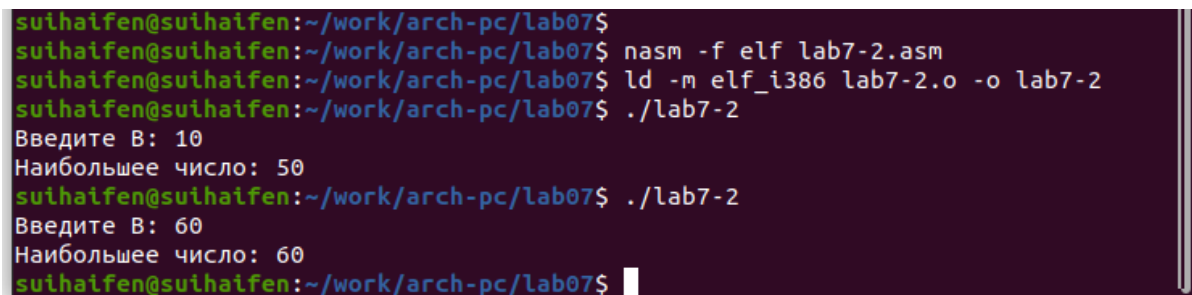
Использование инструкции `jmp` приводит к переходу в любом случае. Однако часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры.

Создаю исполняемый файл и проверяю его работу для разных значений В (рис. 2.8) (рис. 2.9).



```
lab7-2.asm
~/work/arch-pc/lab07
Open Save
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
Matlab Tab Width: 8 Ln 1, Col 1 INS
```

Рис. 2.8: Программа lab7-2.asm



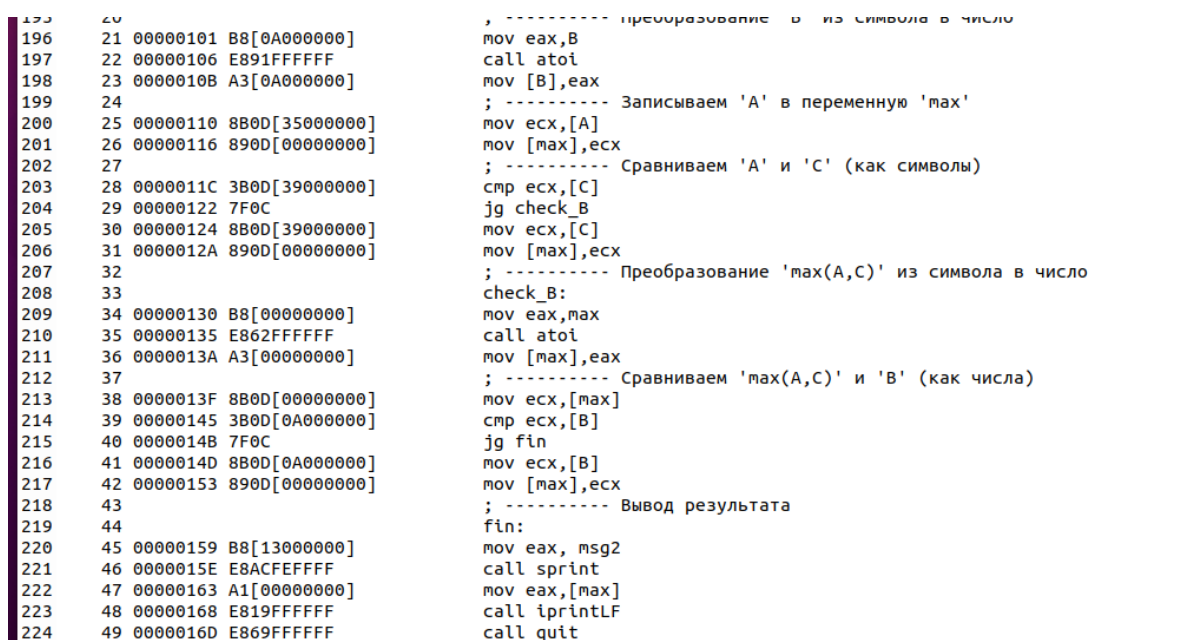
```
suihaifen@suihaifen:~/work/arch-pc/lab07$
suihaifen@suihaifen:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
suihaifen@suihaifen:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
suihaifen@suihaifen:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
suihaifen@suihaifen:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
suihaifen@suihaifen:~/work/arch-pc/lab07$
```

Рис. 2.9: Запуск программы lab7-2.asm

2.2 Изучение структуры файла листинга

Обычно `nasm` создает в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ `-l` и задав имя файла листинга в командной строке.

Создаю файл листинга для программы из файла `lab7-2.asm` (рис. 2.10)



```
196 21 00000101 B8[0A000000]      , ----- преобразование B из символа в число
197 22 00000106 E891FFFFFF      mov eax,B
198 23 0000010B A3[0A000000]      call atoi
199 24                               mov [B],eax
200 25 00000110 8B0D[35000000]      ; ----- Записываем 'A' в переменную 'max'
201 26 00000116 890D[00000000]      mov ecx,[A]
202 27                               mov [max],ecx
203 28 0000011C 3B0D[39000000]      ; ----- Сравниваем 'A' и 'C' (как символы)
204 29 00000122 7F0C              cmp ecx,[C]
205 30 00000124 8B0D[39000000]      jg check_B
206 31 0000012A 890D[00000000]      mov ecx,[C]
207 32                               mov [max],ecx
208 33                               ; ----- Преобразование 'max(A,C)' из символа в число
209 34 00000130 B8[00000000]      check_B:
210 35 00000135 E862FFFFFF      mov eax,max
211 36 0000013A A3[00000000]      call atoi
212 37                               mov [max],eax
213 38 0000013F 8B0D[00000000]      ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
214 39 00000145 3B0D[0A000000]      mov ecx,[max]
215 40 0000014B 7F0C              cmp ecx,[B]
216 41 0000014D 8B0D[0A000000]      jg fin
217 42 00000153 890D[00000000]      mov ecx,[B]
218 43                               mov [max],ecx
219 44                               ; ----- Вывод результата
220 45 00000159 B8[13000000]      fin:
221 46 0000015E E8ACFFFFFF      mov eax,msg2
222 47 00000163 A1[00000000]      call sprintf
223 48 00000168 E819FFFFFF      mov eax,[max]
224 49 0000016D E869FFFFFF      call iprintLF
225                               call quit
```

Рис. 2.10: Файл листинга `lab7-2`

Ознакомимся с его форматом и содержанием.

строка 211

- 34 - номер строки
- 0000012E - адрес
- B8[00000000] - машинный код
- `mov eax,max` - код программы

строка 212

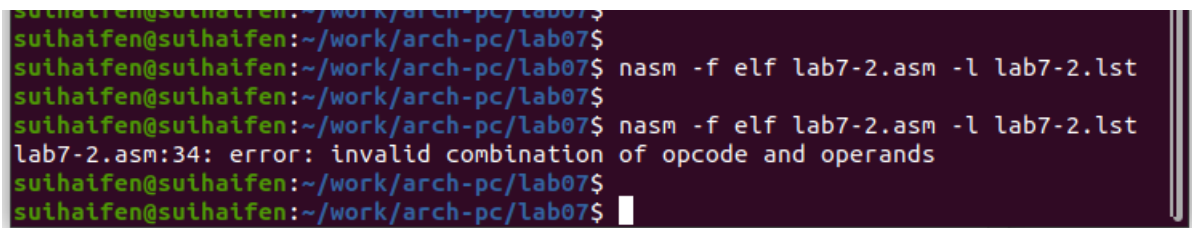
- 35 - номер строки

- 00000133 - адрес
- E864FFFFFF - машинный код
- call atoi - код программы

строка 213

- 36 - номер строки
- 00000138 - адрес
- A3[00000000] - машинный код
- mov [max],eax - код программы

Открываю файл с программой lab7-2.asm и в инструкции с двумя операндами удаляю один операнд. Выполняю трансляцию с получением файла листинга. (рис. 2.11) (рис. 2.12)



```

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

```

Рис. 2.11: Ошибка трансляции lab7-2

```

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 ***** error: invalid combination of opcode and operands
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 iprintlnLF
225 49 00000168 E86EFFFFFF call quit

```

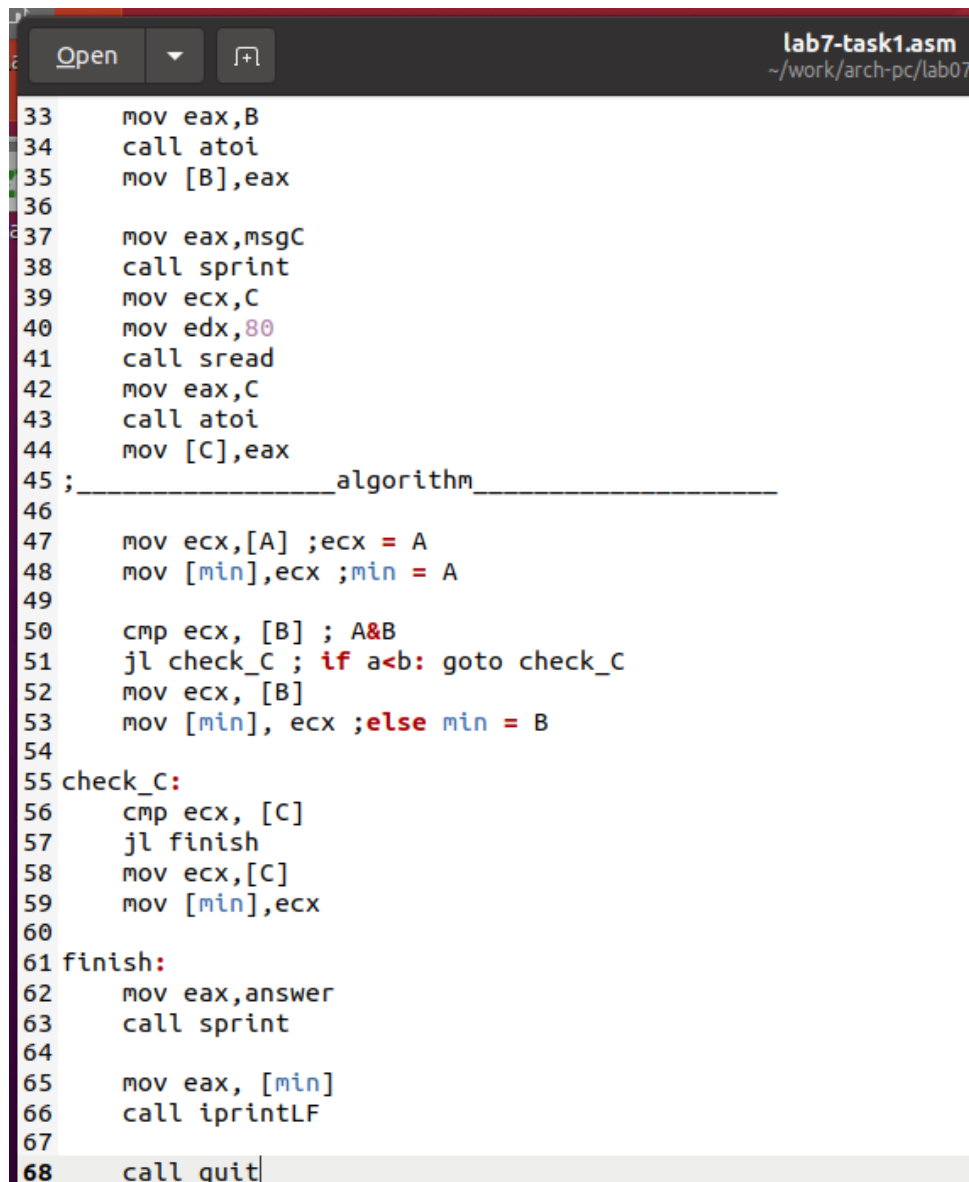
Рис. 2.12: Файл листинга с ошибкой lab7-2

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

2.3 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)

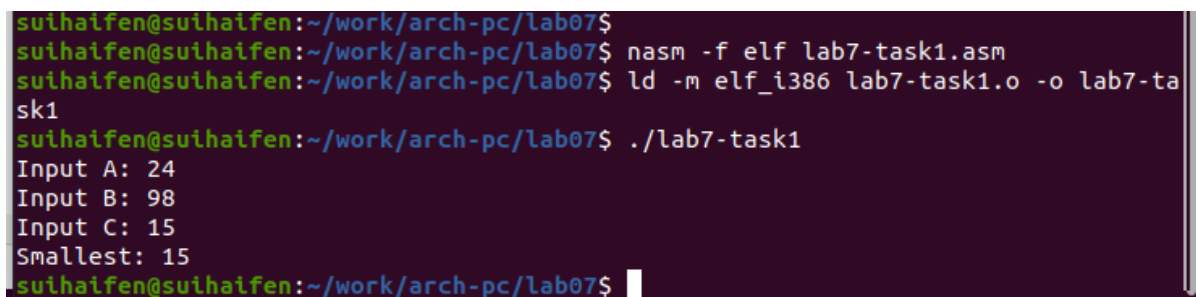
для варианта 9 - 24,98,15



```
lab7-task1.asm
~/work/arch-pc/lab07

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 iprintLF
67
68     call quit
```

Рис. 2.13: Программа lab7-task1.asm



```
suihaifen@suihaifen:~/work/arch-pc/lab07$
suihaifen@suihaifen:~/work/arch-pc/lab07$ nasm -f elf lab7-task1.asm
suihaifen@suihaifen:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-task1.o -o lab7-task1
suihaifen@suihaifen:~/work/arch-pc/lab07$ ./lab7-task1
Input A: 24
Input B: 98
Input C: 15
Smallest: 15
suihaifen@suihaifen:~/work/arch-pc/lab07$
```

Рис. 2.14: Запуск программы lab7-task1.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 9

$$\begin{cases} a + x, x \leq a \\ a, x > a \end{cases}$$

При $x = 5, a = 7$ получается 12.

При $x = 6, a = 4$ получается 4.

```

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

```

Рис. 2.15: Программа lab7-task2.asm

```
suihaifen@suihaifen:~/work/arch-pc/lab07$  
suihaifen@suihaifen:~/work/arch-pc/lab07$ nasm -f elf lab7-task2.asm  
suihaifen@suihaifen:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-task2.o -o lab7-task2  
suihaifen@suihaifen:~/work/arch-pc/lab07$ ./lab7-task2  
Input A: 7  
Input X: 5  
12  
suihaifen@suihaifen:~/work/arch-pc/lab07$  
suihaifen@suihaifen:~/work/arch-pc/lab07$ ./lab7-task2  
Input A: 4  
Input X: 6  
4  
suihaifen@suihaifen:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы lab7-task2.asm

3 Выводы

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