

Отчёт по лабораторной работе 7

Архитектура компьютера

Ушаков Данила Алексеевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файлы листинга	13
2.3	Задание для самостоятельной работы	15
3	Выводы	19

Список иллюстраций

2.1	Код программы lab7-1.asm	7
2.2	Компиляция и запуск программы lab7-1.asm	7
2.3	Код программы lab7-1.asm	9
2.4	Компиляция и запуск программы lab7-1.asm	9
2.5	Код программы lab7-1.asm	10
2.6	Компиляция и запуск программы lab7-1.asm	11
2.7	Код программы lab7-2.asm	12
2.8	Компиляция и запуск программы lab7-2.asm	12
2.9	Файл листинга lab7-2	13
2.10	Ошибка трансляции lab7-2	14
2.11	Файл листинга с ошибкой lab7-2	15
2.12	Код программы prog-1.asm	16
2.13	Компиляция и запуск программы prog-1.asm	16
2.14	Код программы prog-2.asm	18
2.15	Компиляция и запуск программы prog-2.asm	18

Список таблиц

1 Цель работы

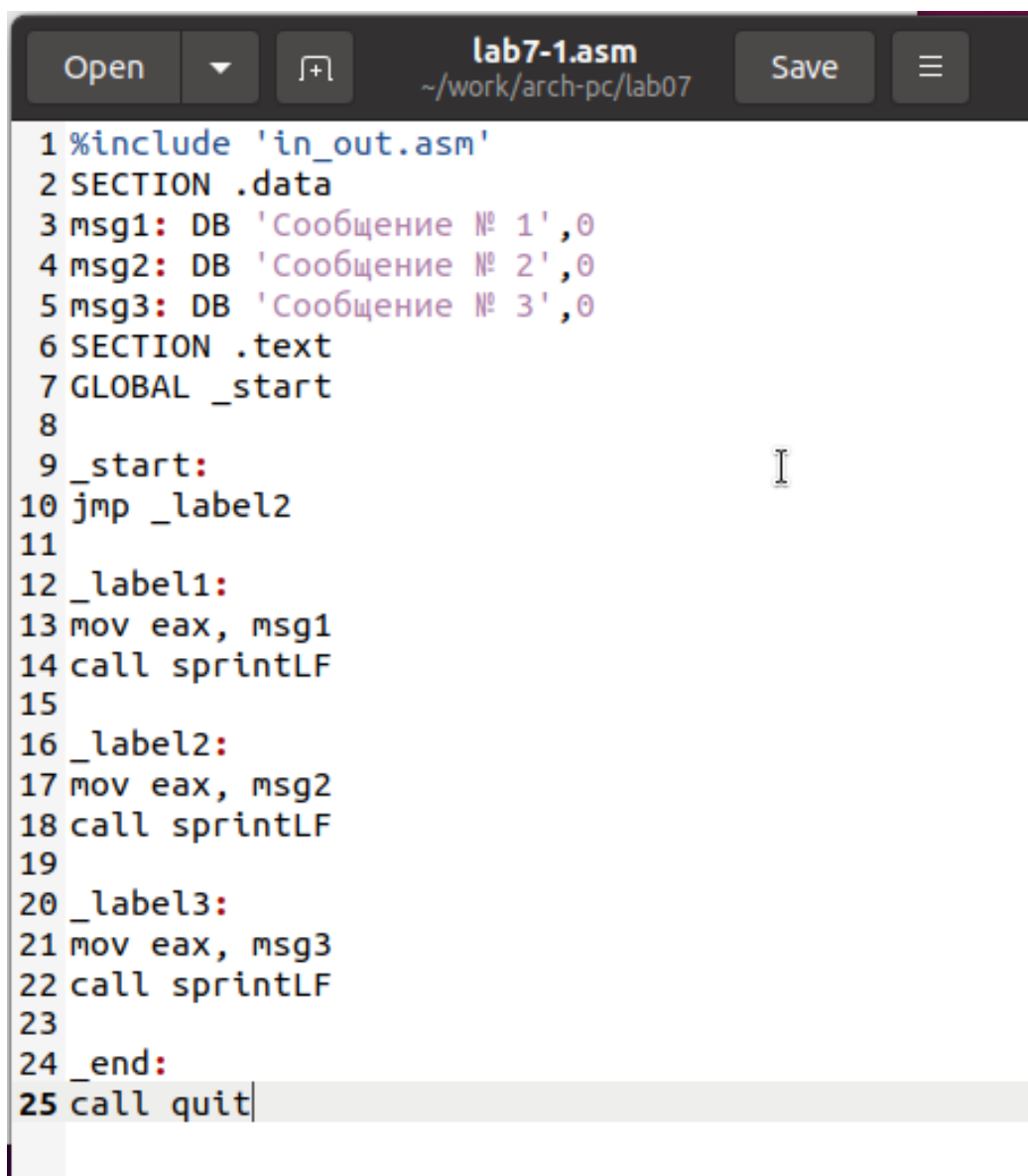
Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

2.1 Реализация переходов в NASM

Я создал каталог для программ лабораторной работы № 7 и файл lab7-1.asm.

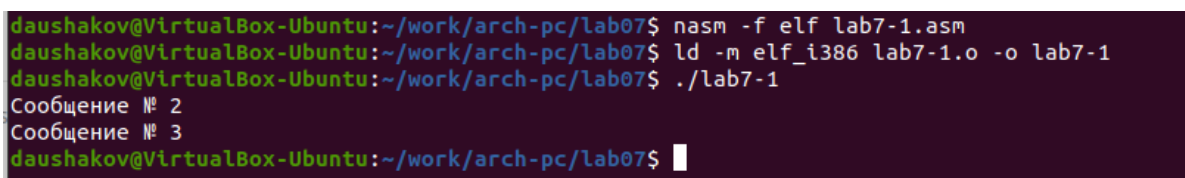
Инструкция `jmp` в NASM используется для выполнения безусловных переходов. Рассмотрим пример программы, в которой используется инструкция `jmp`. Написал текст программы из листинга 7.1 в файле lab7-1.asm. (рис. 2.1)



```
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 _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. 2.1: Код программы lab7-1.asm

Создал исполняемый файл и запустил его. (рис. 2.2)

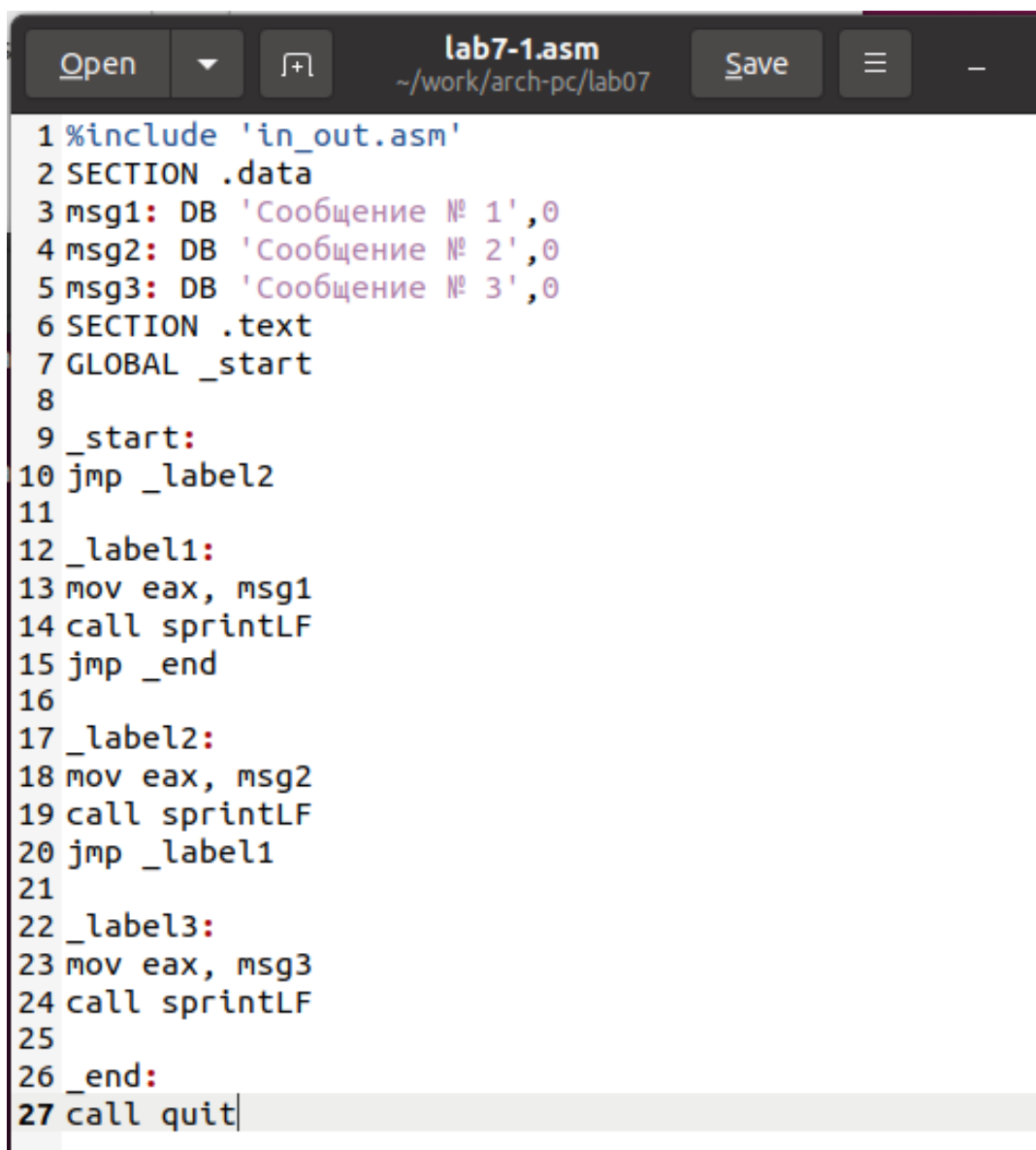


```
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.2: Компиляция и запуск программы lab7-1.asm

Инструкция `jmp` позволяет осуществлять переходы не только вперед, но и назад. Мы изменим программу так, чтобы она сначала выводила “Сообщение № 2”, затем “Сообщение № 1” и завершала работу. Для этого мы добавим в текст программы после вывода “Сообщения № 2” инструкцию `jmp` с меткой `_label1` (чтобы перейти к инструкциям вывода “Сообщения № 1”) и после вывода “Сообщения № 1” добавим инструкцию `jmp` с меткой `_end` (чтобы перейти к инструкции `call quit`).

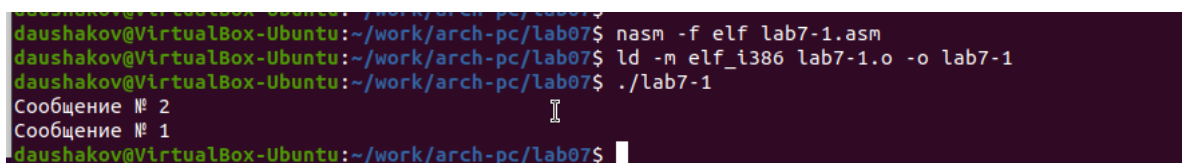
Изменил текст программы в соответствии с листингом 7.2. (рис. 2.3 2.4)



```
lab7-1.asm
~/work/arch-pc/lab07

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. 2.3: Код программы lab7-1.asm



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

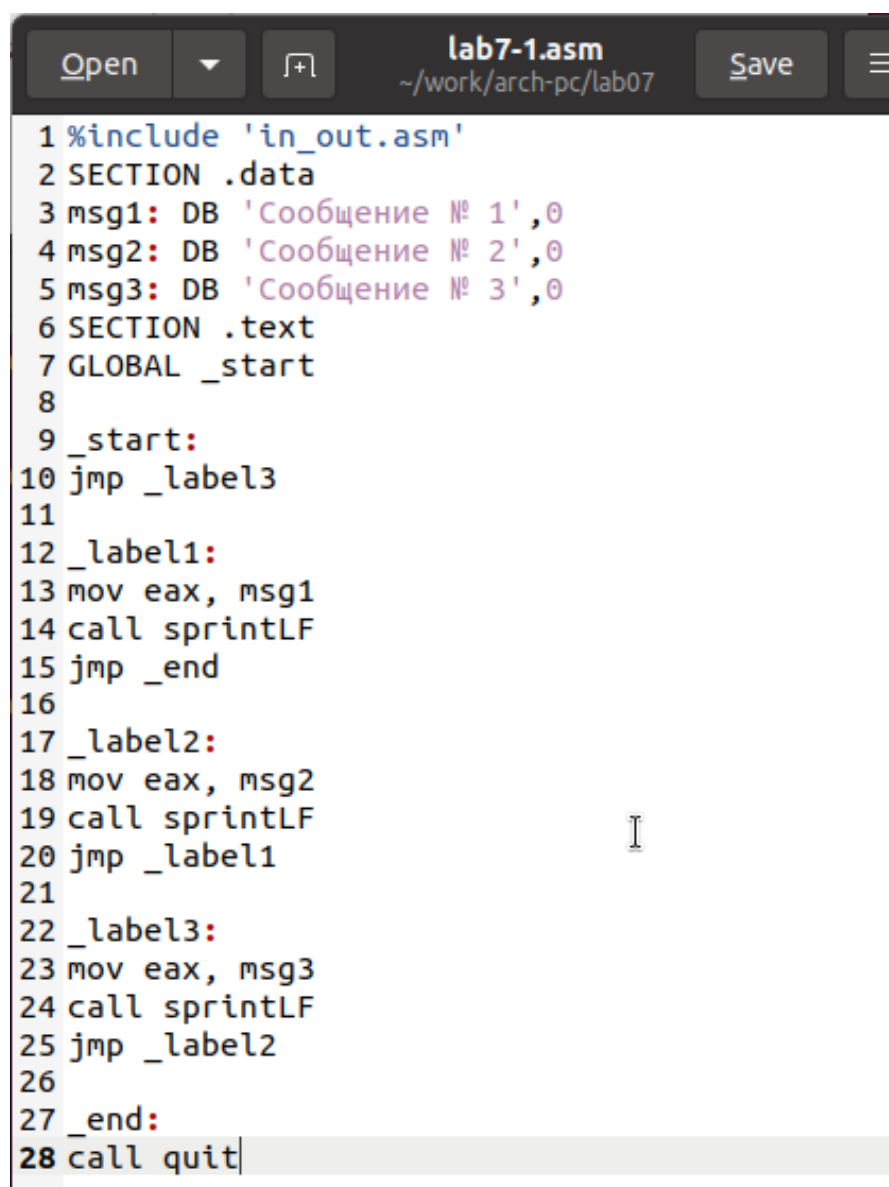
Рис. 2.4: Компиляция и запуск программы lab7-1.asm

Изменил текст программы (рис. 2.5 2.6), изменив инструкции `jmp`, чтобы вывод программы был следующим:

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. 2.5: Код программы lab7-1.asm

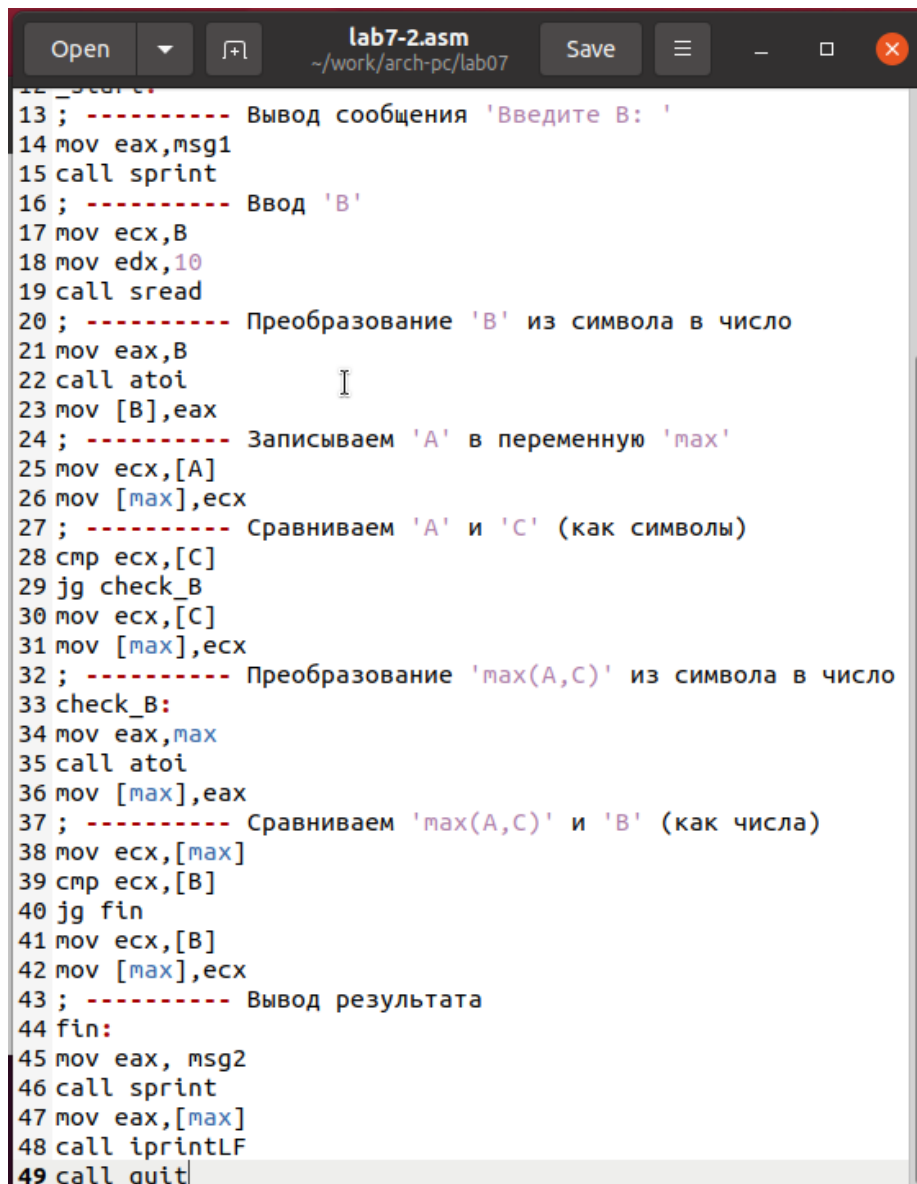
```
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$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

Рис. 2.6: Компиляция и запуск программы lab7-1.asm

Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, то есть переход должен происходить, если выполнено какое-либо условие.

Давайте рассмотрим программу, которая определяет и выводит на экран наибольшую из трех целочисленных переменных: А, В и С. Значения для А и С задаются в программе, а значение В вводится с клавиатуры.

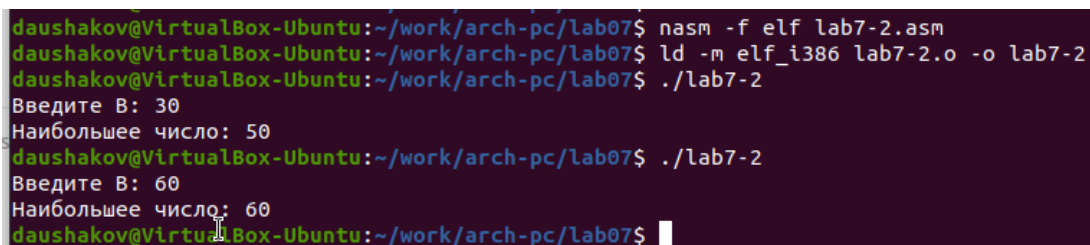
Создал исполняемый файл и проверил его работу для разных значений В. (рис. 2.7 2.8)



```
lab7-2.asm
~/work/arch-pc/lab07
Open Save

12 ; ----- Вывод сообщения 'Введите B: '
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. 2.7: Код программы lab7-2.asm



```
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 30
Наибольшее число: 50
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.8: Компиляция и запуск программы lab7-2.asm

2.2 Изучение структуры файлы листинга

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. 2.9)

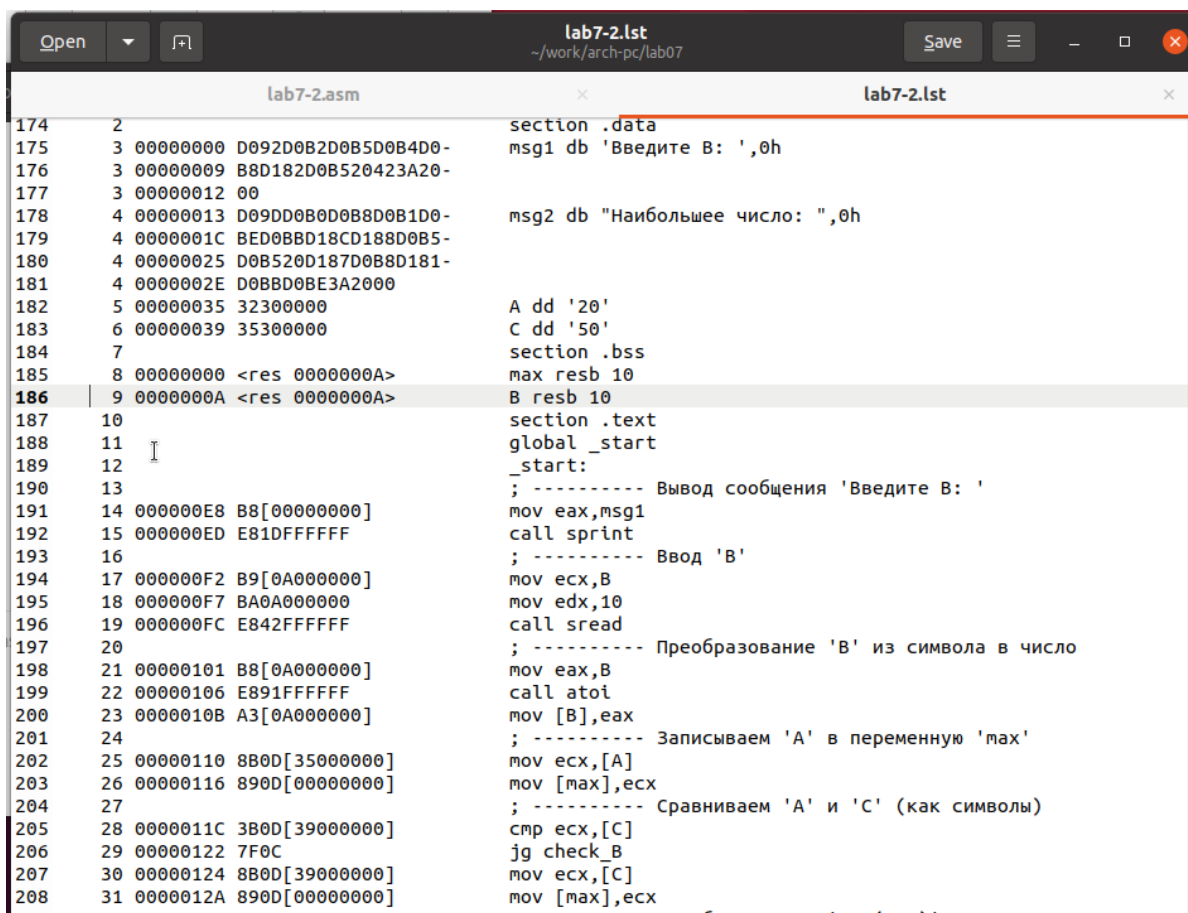


Рис. 2.9: Файл листинга lab7-2

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

строка 194

- 17 - номер строки в подпрограмме
- 000000F2 - адрес

- B9[0A000000] - машинный код
- mov ecx,B - код программы - копирует B в ecx

строка 195

- 18 - номер строки в подпрограмме
- 000000F7 - адрес
- BA0A000000 - машинный код
- mov edx,10 - код программы - копирует 10 в edx

строка 196

- 19 - номер строки в подпрограмме
- 000000FC - адрес
- E842FFFFFF - машинный код
- call sread - код программы - вызов подпрограммы чтения

Открыл файл с программой lab7-2.asm и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга. (рис. 2.10) (рис. 2.11)

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

Рис. 2.10: Ошибка трансляции lab7-2

```

lab7-2.asm
193 16 ; ----- Ввод 'B'
194 17 000000F2 B9[0A000000] mov ecx,B
195 18 000000F7 BA0A000000 mov edx,10
196 19 000000FC E842FFFFFF call sread
197 20 ; ----- Преобразование 'B' из символа в число
198 21 00000101 B8[0A000000] mov eax,B
199 22 00000106 E891FFFFFF call atoi
200 23 0000010B A3[0A000000] mov [B],eax
201 24 ; ----- Записываем 'A' в переменную 'max'
202 25 00000110 8B0D[35000000] mov ecx,[A]
203 26 00000116 890D[00000000] mov [max],ecx
204 27 ; ----- Сравниваем 'A' и 'C' (как символы)
205 28 0000011C 3B0D[39000000] cmp ecx,[C]
206 29 00000122 7F0C jg check_B
207 30 00000124 8B0D[39000000] mov ecx,[C]
208 31 0000012A 890D[00000000] mov [max],ecx
209 32 ; ----- Преобразование 'max(A,C)' из символа в число
210 33 check_B:
211 34 mov eax,
212 34 *****
213 35 00000130 E867FFFFFF call atoi
214 36 00000135 A3[00000000] mov [max],eax
215 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
216 38 0000013A 8B0D[00000000] mov ecx,[max]
217 39 00000140 3B0D[0A000000] cmp ecx,[B]
218 40 00000146 7F0C jg fin
219 41 00000148 8B0D[0A000000] mov ecx,[B]
220 42 0000014E 890D[00000000] mov [max],ecx
221 43 ; ----- Вывод результата
222 44 fin:
223 45 00000154 B8[13000000] mov eax,msg2
224 46 00000159 E8B1FEFFFF call sprint
225 47 0000015E A1[00000000] mov eax,[max]
226 48 00000163 E81EFFFFFF call iprintLF
227 49 00000168 F86FFFFFFF call quit

```

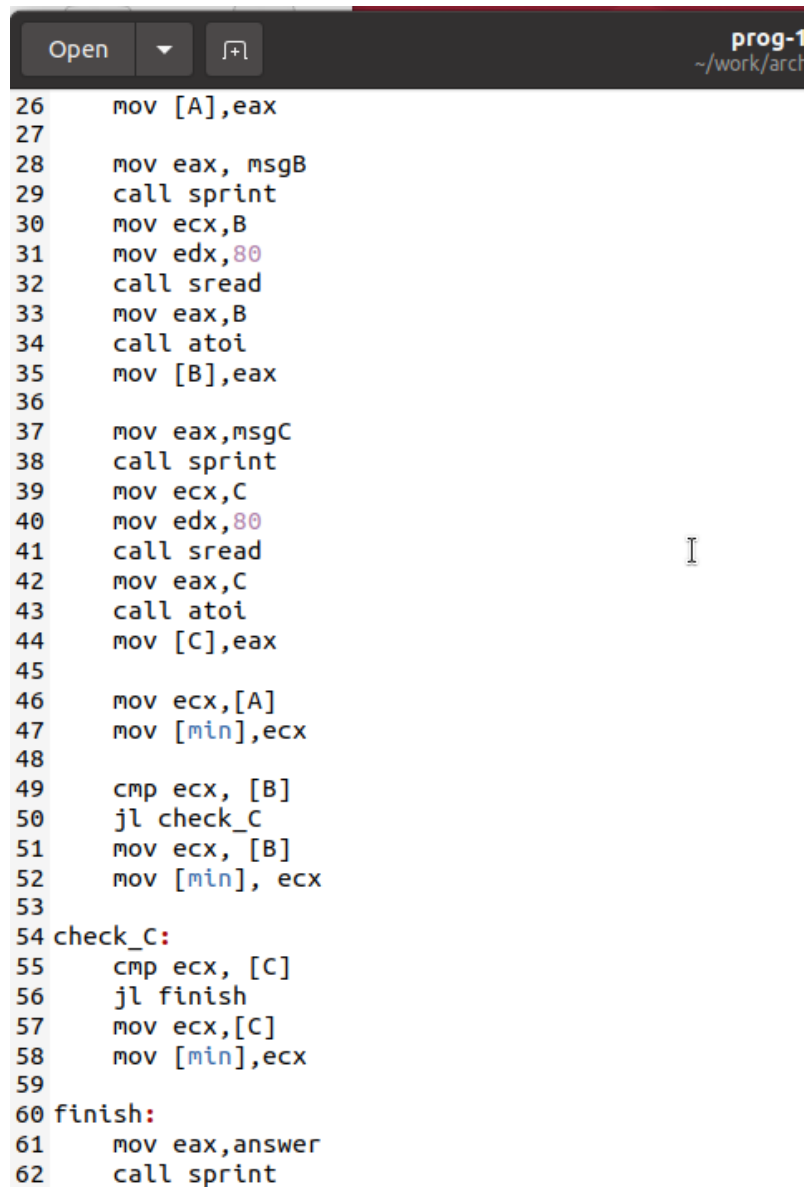
Рис. 2.11: Файл листинга с ошибкой lab7-2

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

2.3 Задание для самостоятельной работы

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. 2.12) (рис. 2.13)

Мой вариант 16 - числа: 44,74,17

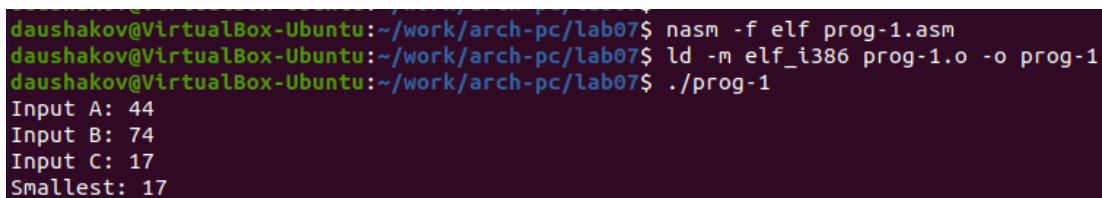


```

26    mov [A],eax
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
46    mov ecx,[A]
47    mov [min],ecx
48
49    cmp ecx, [B]
50    jnl check_C
51    mov ecx, [B]
52    mov [min], ecx
53
54 check_C:
55    cmp ecx, [C]
56    jnl finish
57    mov ecx,[C]
58    mov [min],ecx
59
60 finish:
61    mov eax,answer
62    call sprint

```

Рис. 2.12: Код программы prog-1.asm



```

daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ nasm -f elf prog-1.asm
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 prog-1.o -o prog-1
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ ./prog-1
Input A: 44
Input B: 74
Input C: 17
Smallest: 17

```

Рис. 2.13: Компиляция и запуск программы prog-1.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. 2.14) (рис. 2.15)

Мой вариант 16

$$\begin{cases} x + 4, x < 4 \\ ax, x \geq 4 \end{cases}$$

```

12 GLOBAL _start
13
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 edx, 4
34     mov ebx, [X]
35     cmp ebx, edx
36     jb first
37     jmp second
38
39 first:
40     mov eax,[X]
41     add eax,4
42     call iprintLF
43     call quit
44 second:
45     mov eax,[X]
46     mov ebx,[A]
47     mul ebx
48     call iprintLF
49     call quit

```

Рис. 2.14: Код программы prog-2.asm

```

daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ nasm -f elf prog-2.asm
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 prog-2.o -o prog-2
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ ./prog-2
Input A: 1
Input X: 1
5
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$ ./prog-2
Input A: 1
Input X: 7
7
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab07$

```

Рис. 2.15: Компиляция и запуск программы prog-2.asm

3 Выводы

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