

Отчёт по лабораторной работе 7

Архитектура компьютеров

Довлетов Арслан

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файлы листинга	12
2.3	Задание для самостоятельной работы	14
3	Выводы	18

Список иллюстраций

2.1	Программа в файле lab7-1.asm	7
2.2	Запуск программы lab7-1.asm	7
2.3	Программа в файле lab7-1.asm	8
2.4	Запуск программы lab7-1.asm	9
2.5	Программа в файле lab7-1.asm	10
2.6	Запуск программы lab7-1.asm	10
2.7	Программа в файле lab7-2.asm	11
2.8	Запуск программы lab7-2.asm	12
2.9	Файл листинга lab7-2	12
2.10	Ошибка трансляции lab7-2	14
2.11	Файл листинга с ошибкой lab7-2	14
2.12	Программа в файле task7-1.asm	15
2.13	Запуск программы task7-1.asm	15
2.14	Программа в файле task7-2.asm	17
2.15	Запуск программы task7-2.asm	17

Список таблиц

1 Цель работы

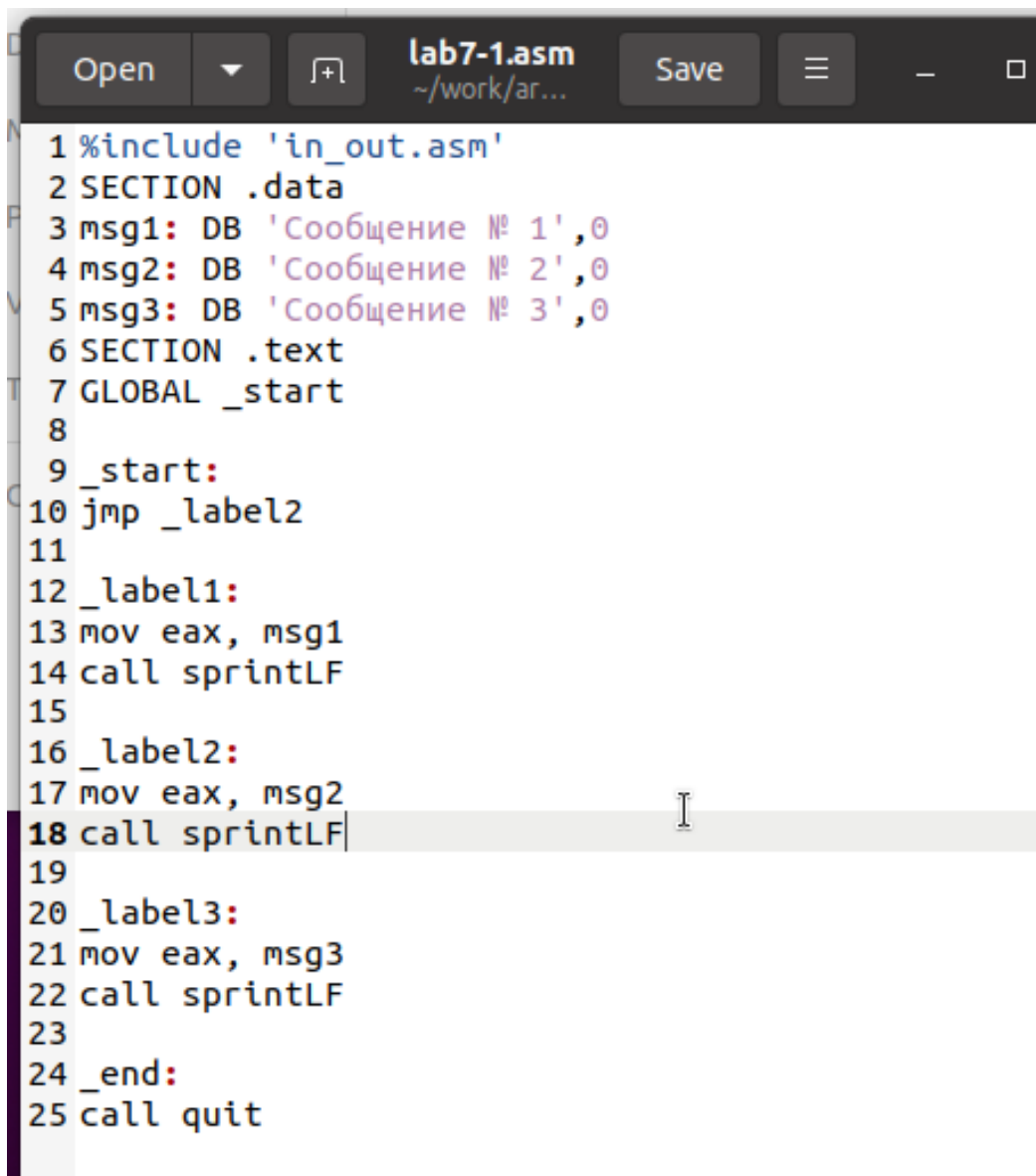
Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

2.1 Реализация переходов в NASM

Создал каталог для программам лабораторной работы № 7 и файл lab7-1.asm

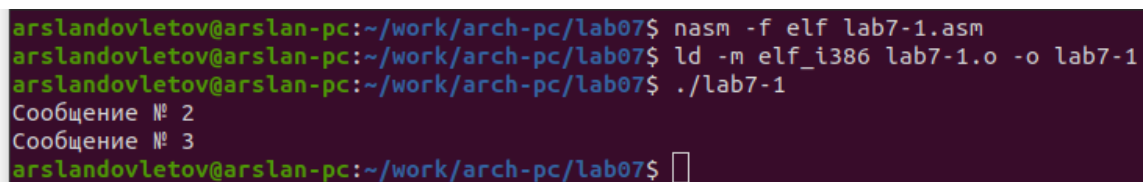
Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Написал в файл lab7-1.asm текст программы из листинга 7.1.



```
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: Программа в файле lab7-1.asm

Создал исполняемый файл и запустил его.

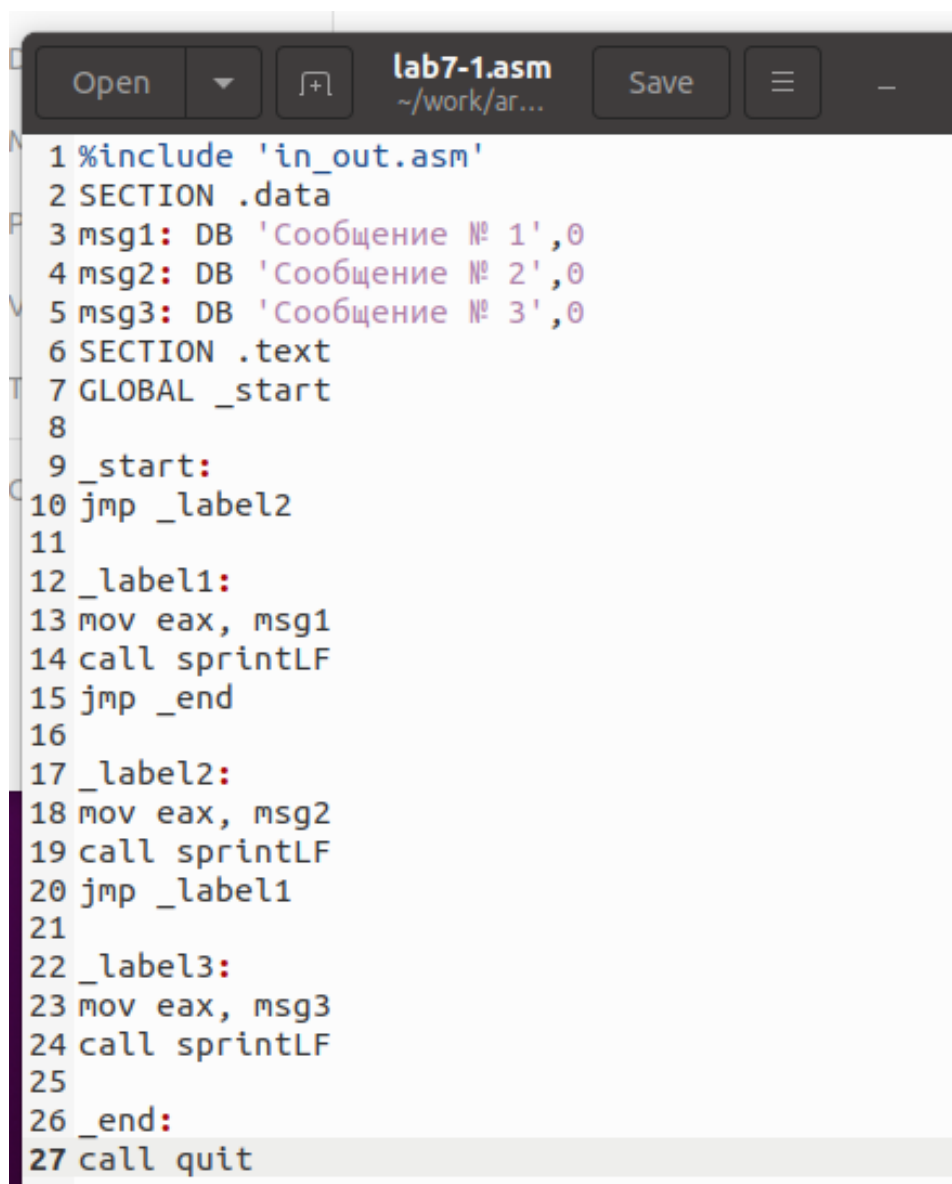


```
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
arslandovletov@arslan-pc:~/work/arch-pc/lab07$
```

Рис. 2.2: 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.



```
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: Программа в файле lab7-1.asm


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

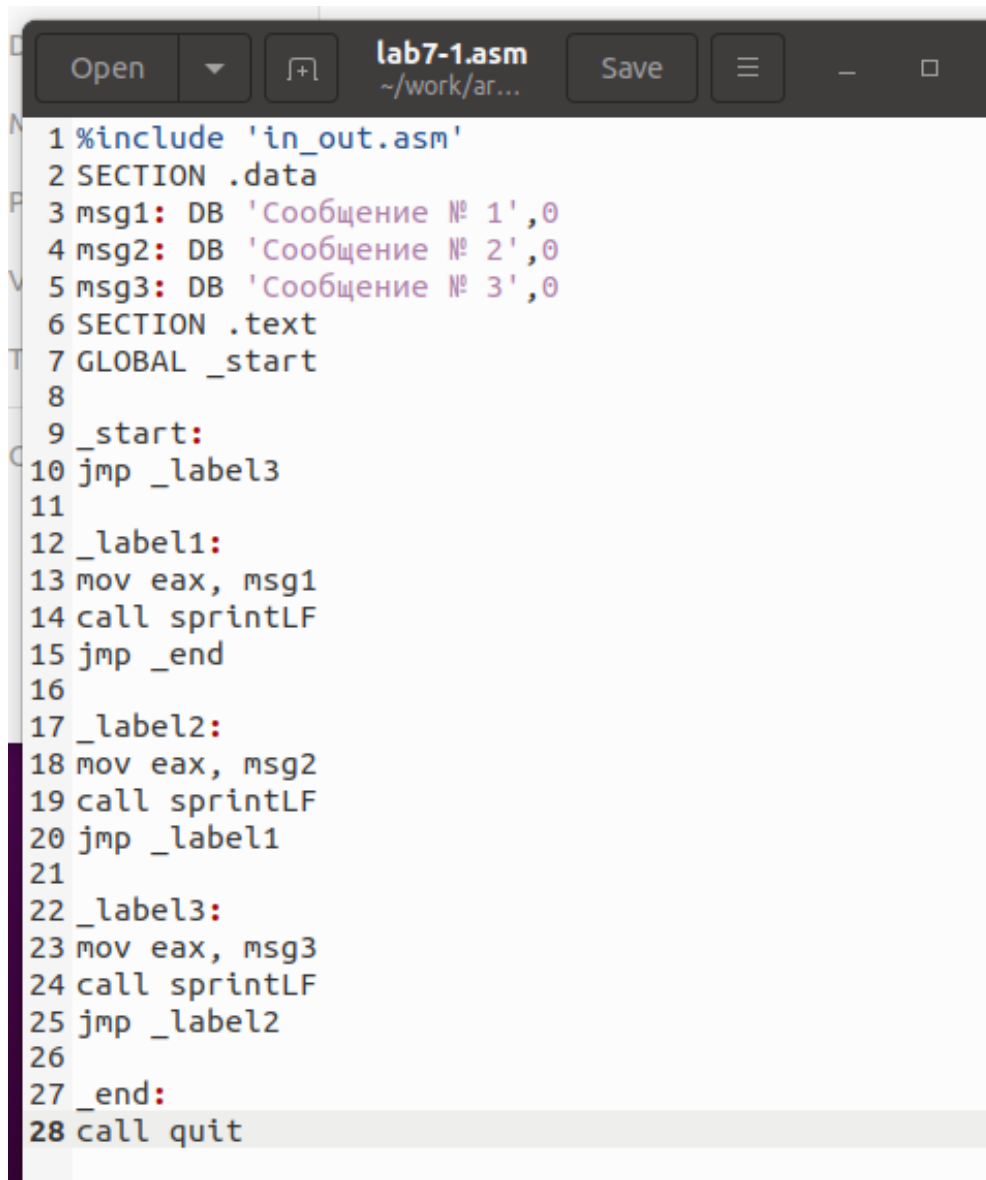
Рис. 2.4: Запуск программы lab7-1.asm

Изменил текст программы, изменив инструкции jmp, чтобы вывод программы был следующим:

Сообщение № 3

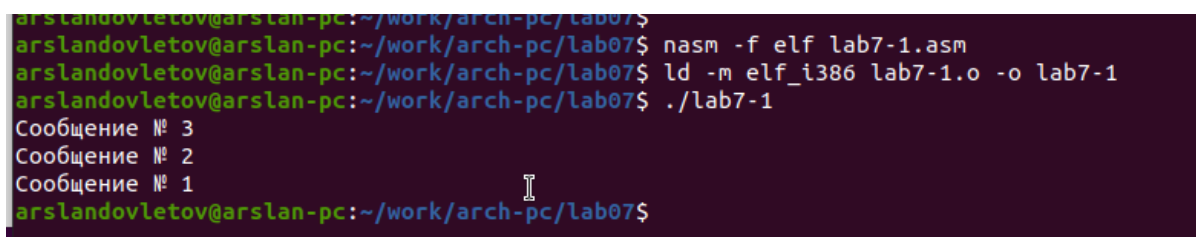
Сообщение № 2

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: Программа в файле lab7-1.asm



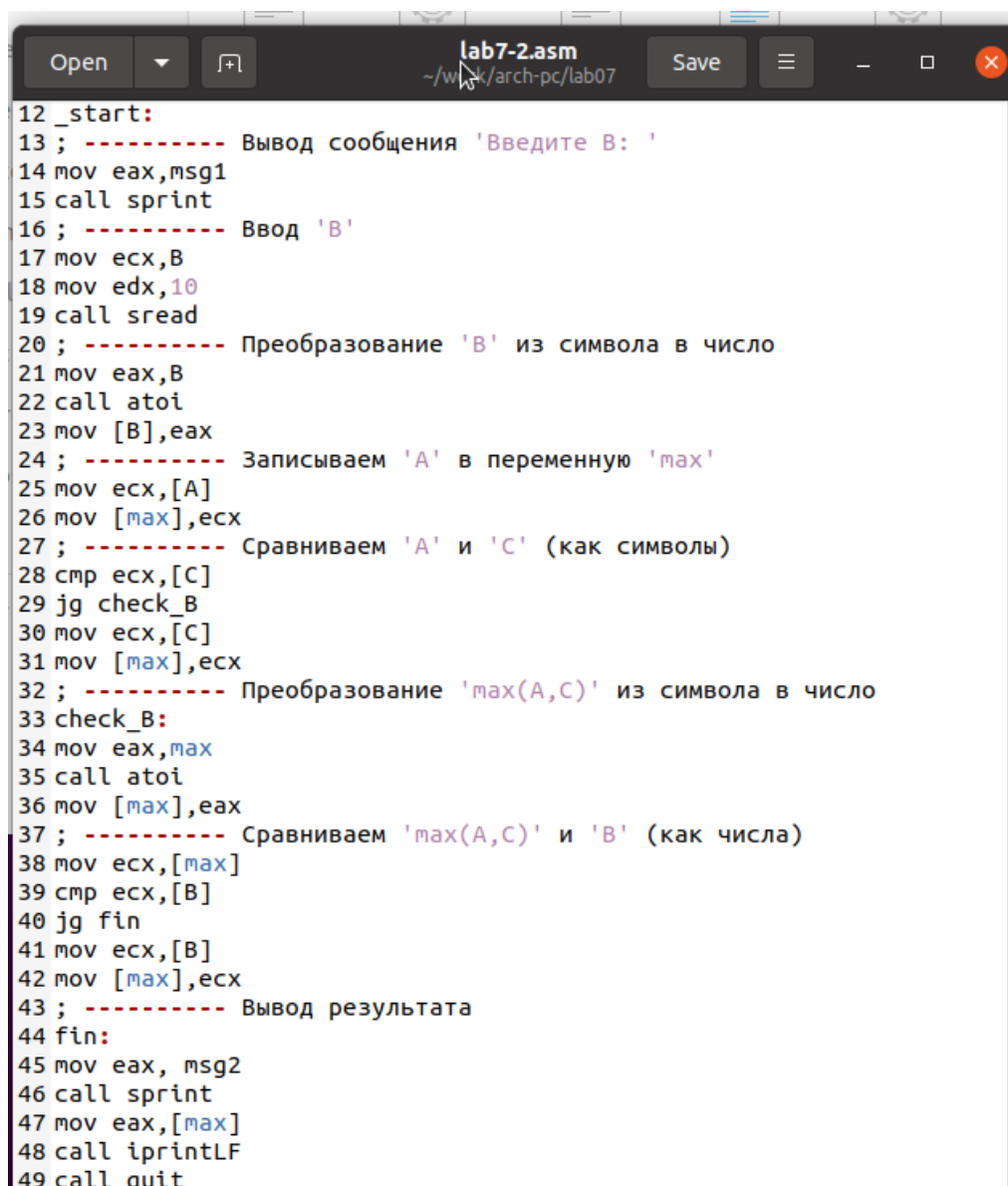
```
arslandovletov@arslan-pc:~/work/arch-pc/lab07$
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
arslandovletov@arslan-pc:~/work/arch-pc/lab07$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

Рис. 2.6: 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.



```
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. 2.7: Программа в файле lab7-2.asm

```

arslandovletov@arslan-pc:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 3
Наибольшее число: 50
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 40
Наибольшее число: 50
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
arslandovletov@arslan-pc:~/work/arch-pc/lab07$

```

Рис. 2.8: 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`

```

183      8 00000000 <res 0000000A>      section .res
184      9 0000000A <res 0000000A>      max resb 10
185     10                                     B resb 10
186     11                                     section .text
187     12                                     global _start
188     13                                     _start:
189     14 000000E8 B8[00000000]          ; ----- Вывод сообщения 'Введите B: '
190     15 000000ED E81DFFFFFF          mov eax,msg1
191     16                                     call sprint
192     17 000000F2 B9[0A000000]          ; ----- Ввод 'B'
193     18 000000F7 BA0A000000          mov ecx,B
194     19 000000FC E842FFFFFF          mov edx,10
195     20                                     call sread
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                                     mov [max],ecx

```

Рис. 2.9: Файл листинга lab7-2

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

строка 203

- 28 - номер строки в подпрограмме
- 0000011C - адрес
- 3B0D[39000000] - машинный код
- cmp esx,[C] - код программы - сравнивает регистр esx и переменную C

строка 204

- 29 - номер строки в подпрограмме
- 00000122 - адрес
- 7F0C - машинный код
- jg check_B - код программы - если >, то переход к метке check_B

строка 205

- 30 - номер строки в подпрограмме
- 00000124 - адрес
- 8B0D[39000000] - машинный код
- mov esx,[C] - код программы - перекладывает в регистр esx значение переменной C

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

```

arslandovletov@arslan-pc:~/work/arch-pc/lab07$
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:30: error: invalid combination of opcode and operands
arslandovletov@arslan-pc:~/work/arch-pc/lab07$

```

Рис. 2.10: Ошибка трансляции lab7-2

```

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 7F06 jg check_B
205 30 mov ecx,
206 30 *****
207 31 00000124 890D[00000000] error: invalid combination of opcode and operands
208 32 mov [max],ecx
209 33 ; ----- Преобразование 'max(A,C)' из символа в число
210 34 0000012A B8[00000000] check_B:
211 35 0000012F E868FFFFFF mov eax,max
212 36 00000134 A3[00000000] call atoi
213 37 mov [max],eax
214 38 00000139 8B0D[00000000] ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
215 39 0000013F 3B0D[0A000000] mov ecx,[max]
216 40 00000145 7F0C cmp ecx,[B]
217 41 00000147 8B0D[0A000000] jg fin
218 42 0000014D 890D[00000000] mov ecx,[B]
219 43 mov [max],ecx
220 44 ; ----- Вывод результата
221 45 00000153 B8[13000000] fin:
222 46 00000158 E8B2FFFFFF mov eax,msg2
223 47 0000015D A1[00000000] call sprint
224 48 00000162 E81FFFFFFF mov eax,[max]
225 49 00000167 E86FFFFFFF call iprintLF
225 49 00000167 E86FFFFFFF call quit

```

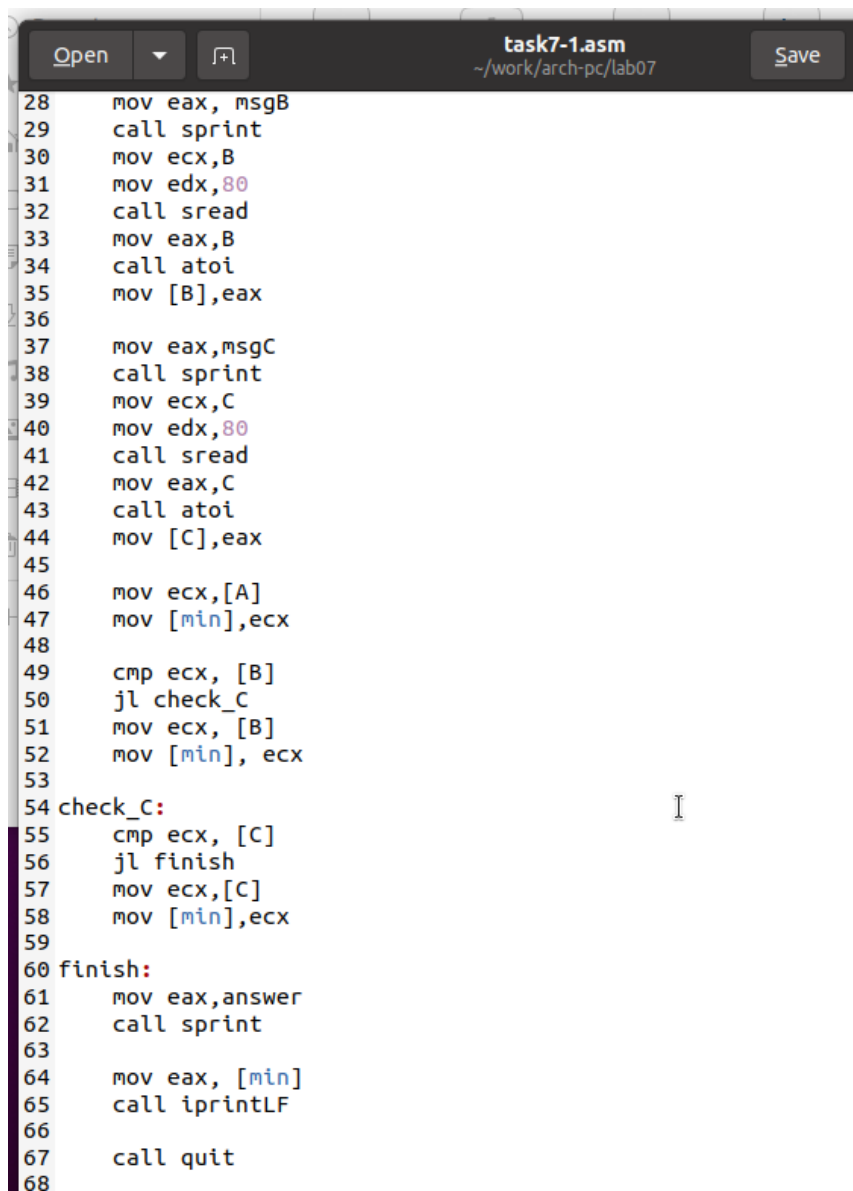
Рис. 2.11: Файл листинга с ошибкой lab7-2

Объектный файл не смог создаваться 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

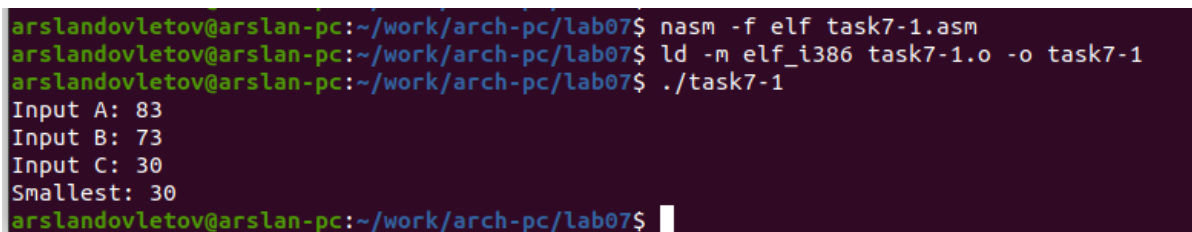
для варианта 18 - 83,73,30



```
task7-1.asm
~/work/arch-pc/lab07

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  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
```

Рис. 2.12: Программа в файле task7-1.asm



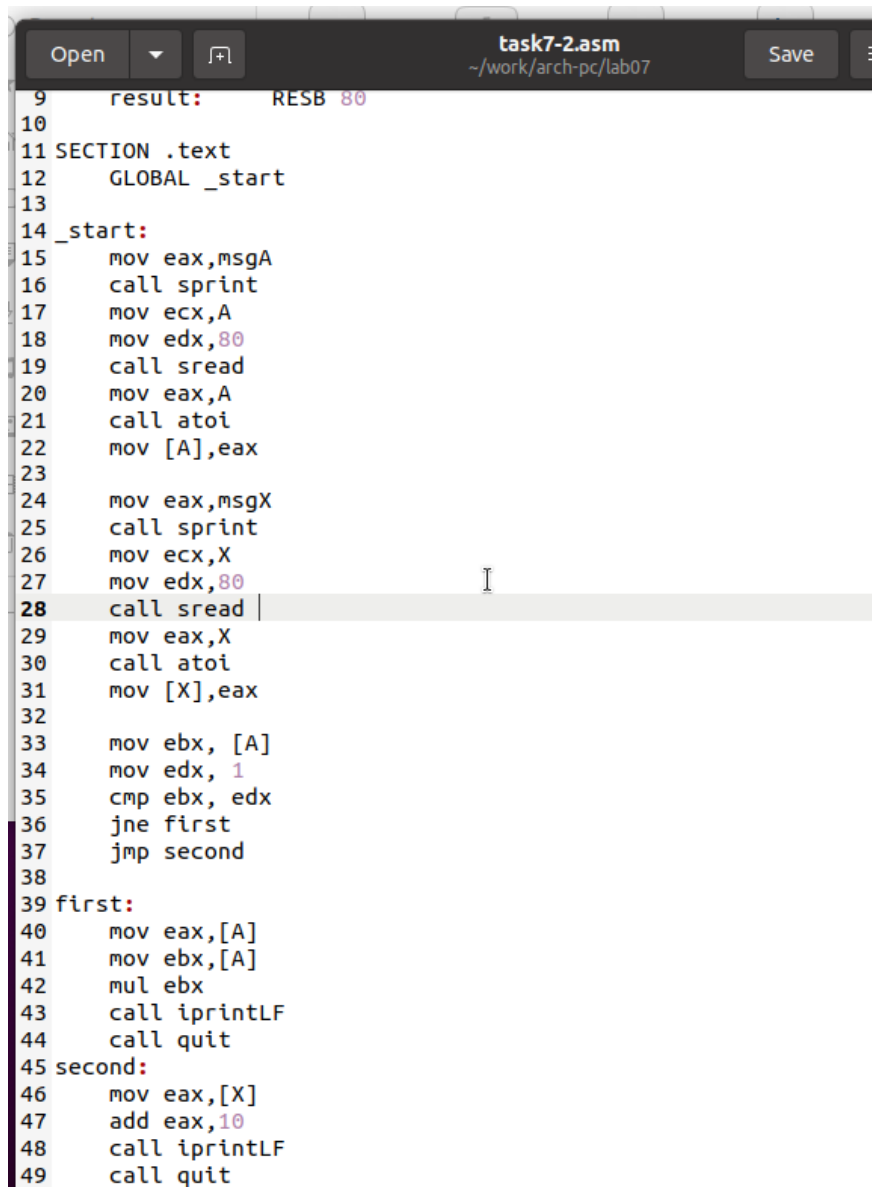
```
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ nasm -f elf task7-1.asm
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ ld -m elf_i386 task7-1.o -o task7-1
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ ./task7-1
Input A: 83
Input B: 73
Input C: 30
Smallest: 30
arslandovletov@arslan-pc:~/work/arch-pc/lab07$
```

Рис. 2.13: Запуск программы task7-1.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.

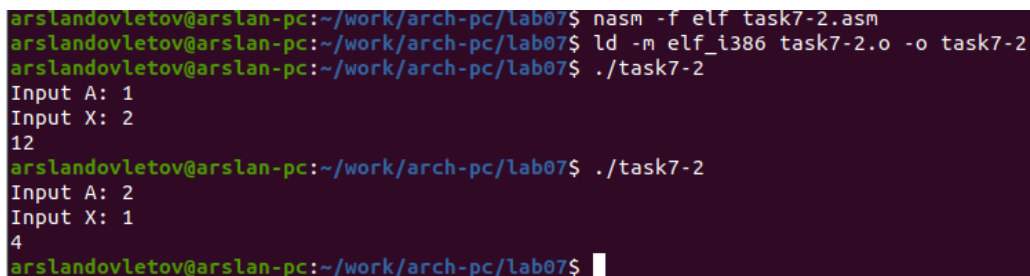
для варианта 18

$$\begin{cases} a^2, a \neq 1 \\ x + 10, a = 1 \end{cases}$$



```
9      result:      RESB 80
10
11 SECTION .text
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 ebx, [A]
34     mov edx, 1
35     cmp ebx, edx
36     jne first
37     jmp second
38
39 first:
40     mov eax,[A]
41     mov ebx,[A]
42     mul ebx
43     call iprintLF
44     call quit
45 second:
46     mov eax,[X]
47     add eax,10
48     call iprintLF
49     call quit
```

Рис. 2.14: Программа в файле task7-2.asm



```
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ nasm -f elf task7-2.asm
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ ld -m elf_i386 task7-2.o -o task7-2
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ ./task7-2
Input A: 1
Input X: 2
12
arslandovletov@arslan-pc:~/work/arch-pc/lab07$ ./task7-2
Input A: 2
Input X: 1
4
arslandovletov@arslan-pc:~/work/arch-pc/lab07$
```

Рис. 2.15: Запуск программы task7-2.asm

3 Выводы

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