

Отчёт по лабораторной работе 7

Архитектура компьютеров

Иван Горбунов

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	20

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:	9
2.4	Запуск программы lab7-1.asm:	9
2.5	Программа в файле lab7-1.asm	10
2.6	Запуск программы lab7-1.asm	11
2.7	Программа в файле lab7-2.asm	12
2.8	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	Программа в файле task.asm	16
2.13	Запуск программы task.asm	17
2.14	Программа в файле task2.asm	18
2.15	Запуск программы task2.asm	19

Список таблиц

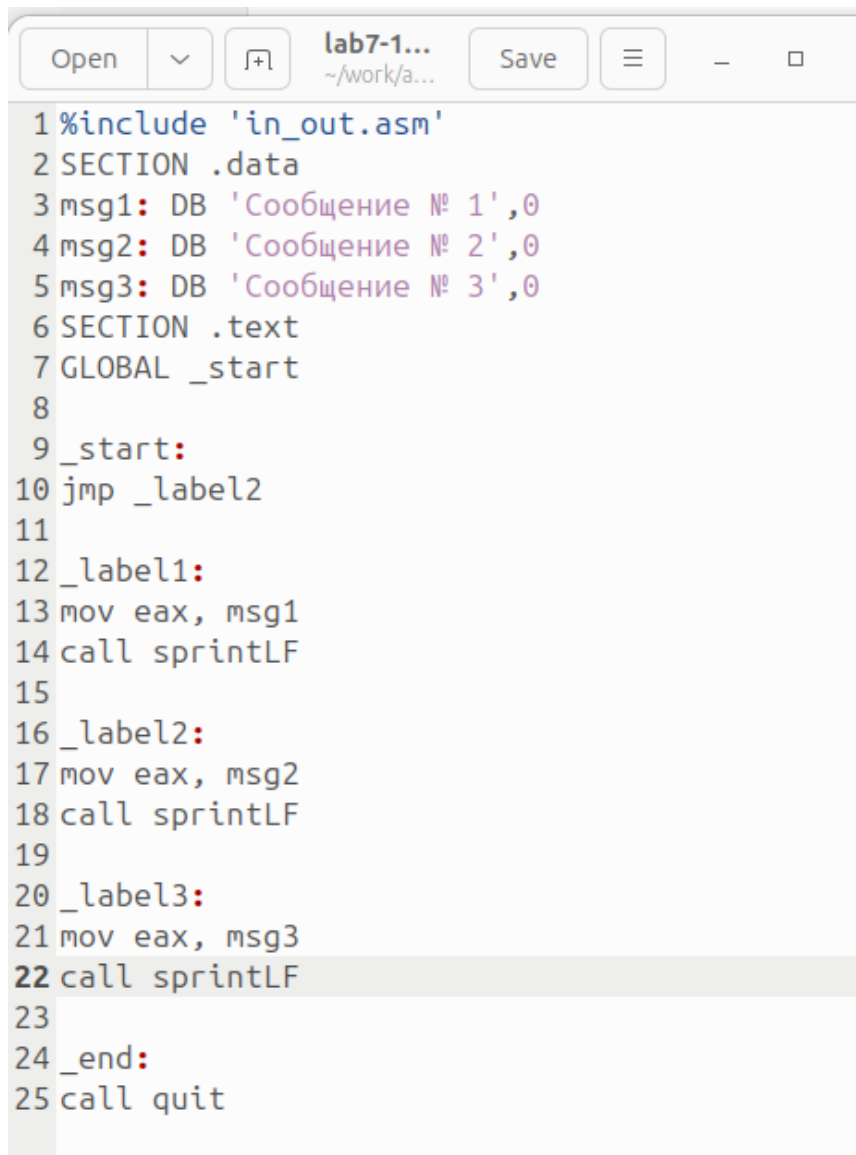
1 Цель работы

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

1. Создал каталог для программ лабораторной работы № 7 и файл lab7-1.asm
2. Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`.

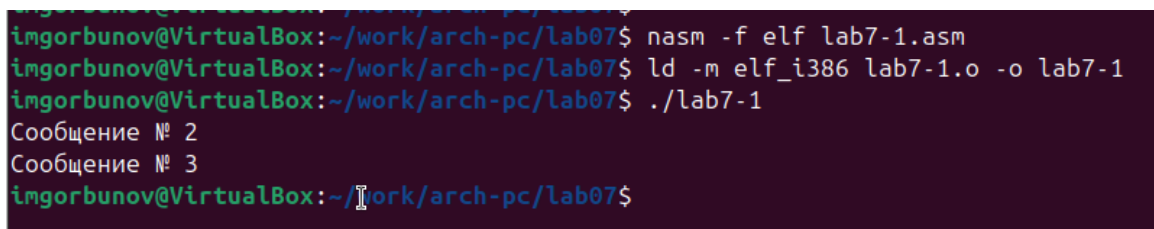
Написал в файл lab7-1.asm текст программы из листинга 7.1.



```
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
```

Рисунок 2.1: Программа в файле lab7-1.asm

Создал исполняемый файл и запустил его.

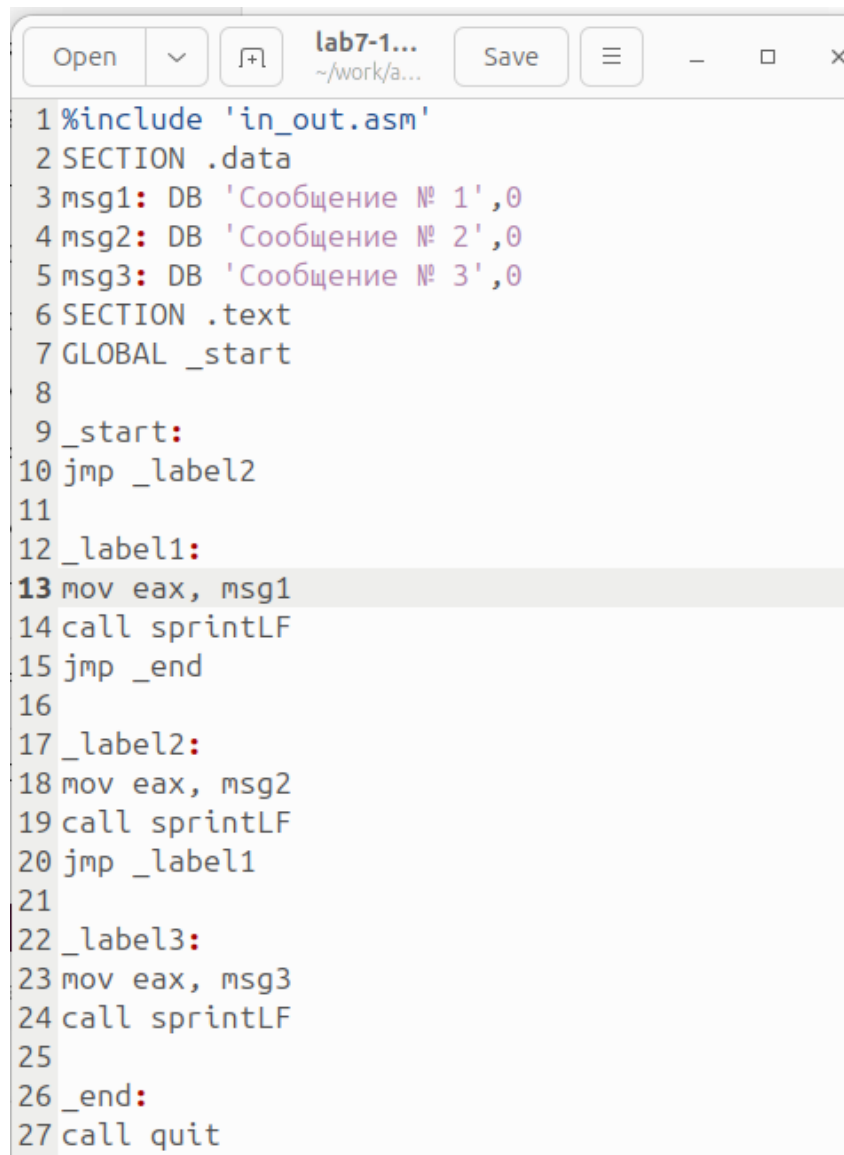


```
imgorbunov@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
imgorbunov@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
imgorbunov@VirtualBox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
imgorbunov@VirtualBox:~/work/arch-pc/lab07$
```

Рисунок 2.2: Запуск программы lab7-1.asm

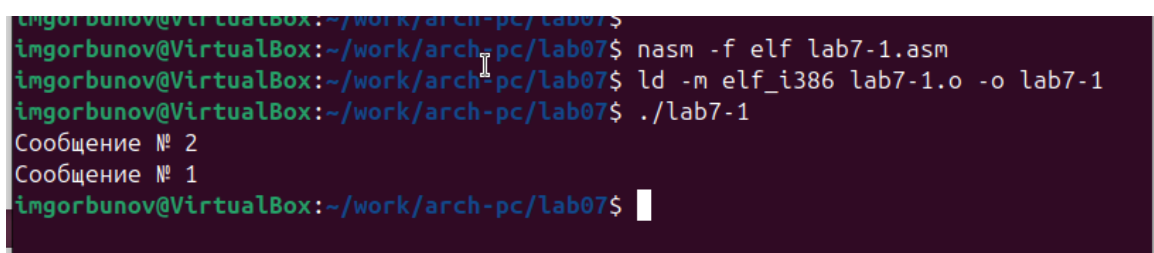
Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала „Сообщение № 2“, потом „Сообщение № 1“ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. 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 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
```

Рисунок 2.3: Программа в файле lab7-1.asm:



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

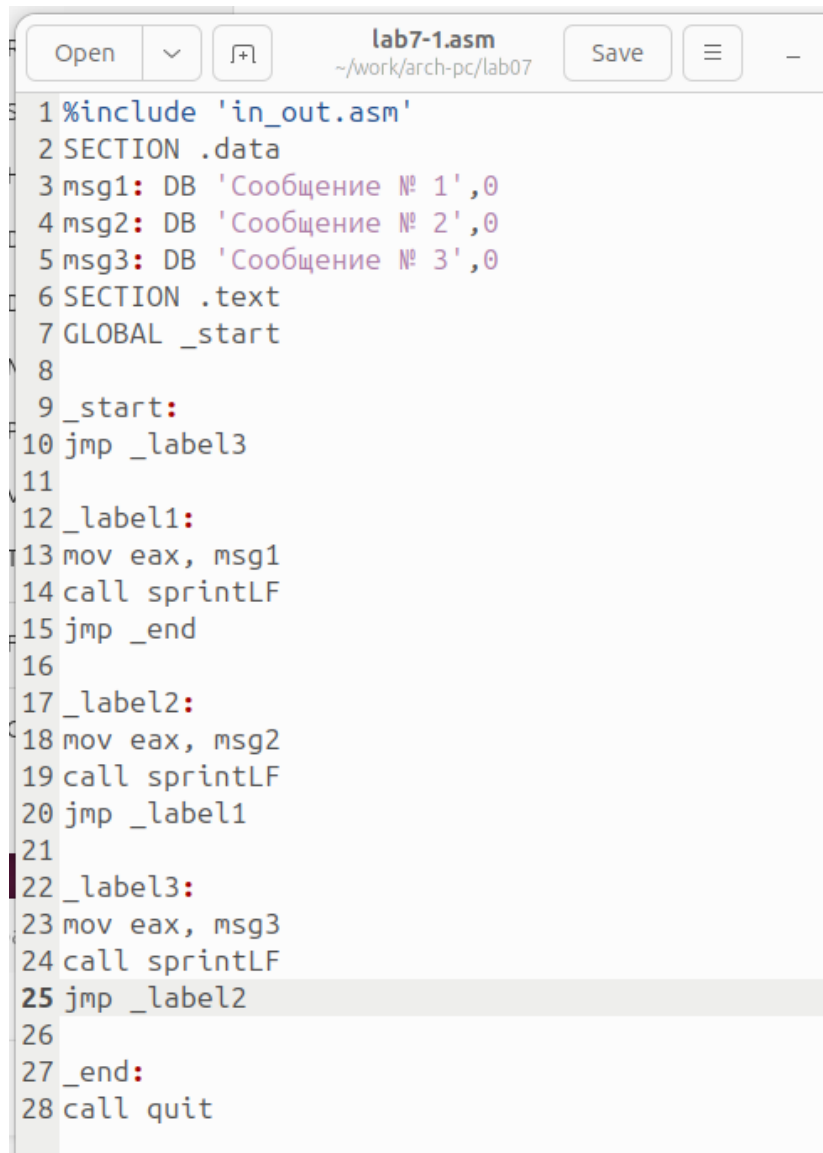
Рисунок 2.4: Запуск программы lab7-1.asm:

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 sprintf
15 jmp _end
16
17 _label2:
18 mov eax, msg2
19 call sprintf
20 jmp _label1
21
22 _label3:
23 mov eax, msg3
24 call sprintf
25 jmp _label2
26
27 _end:
28 call quit
```

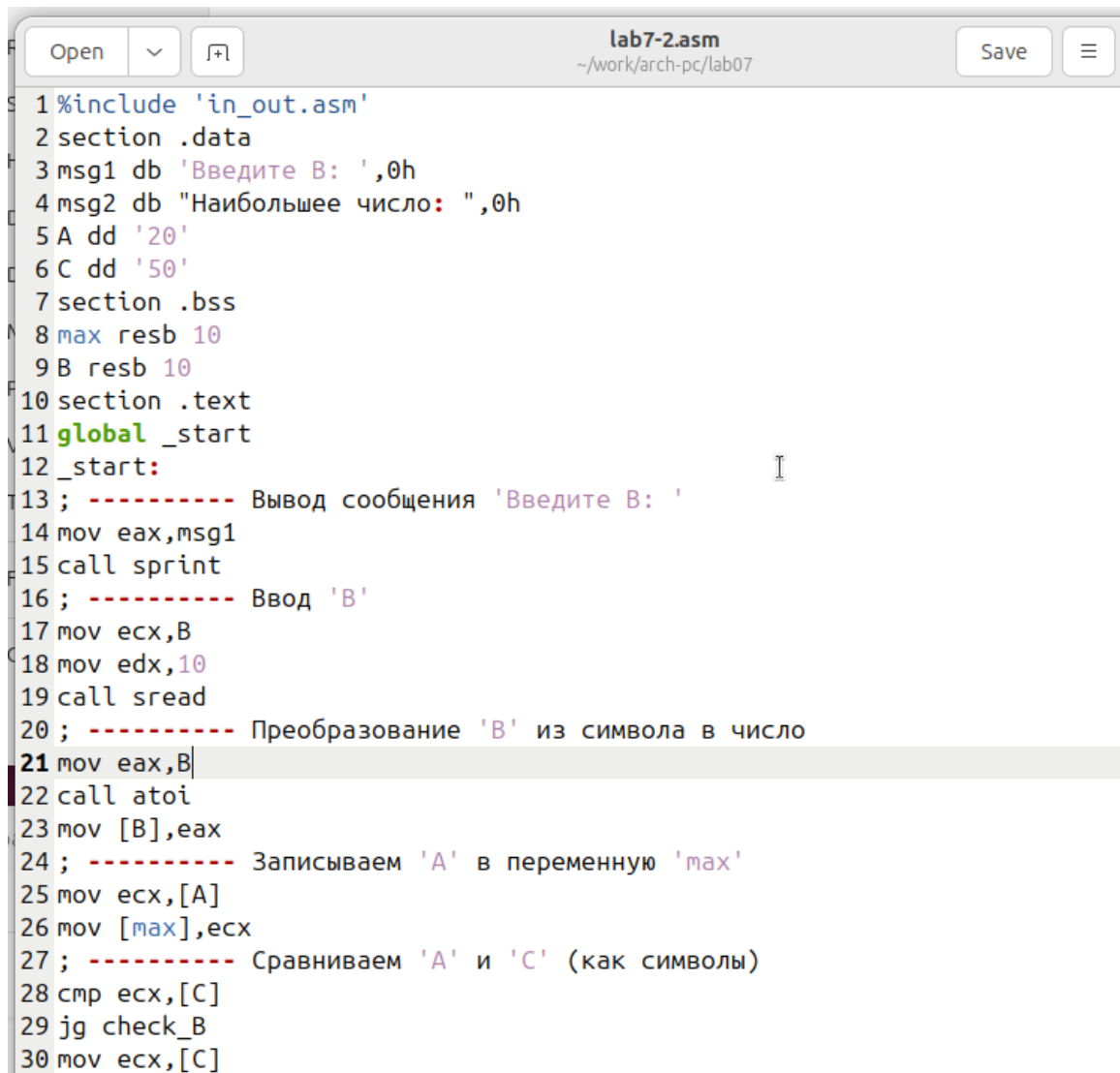
Рисунок 2.5: Программа в файле lab7-1.asm

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

Рисунок 2.6: Запуск программы lab7-1.asm

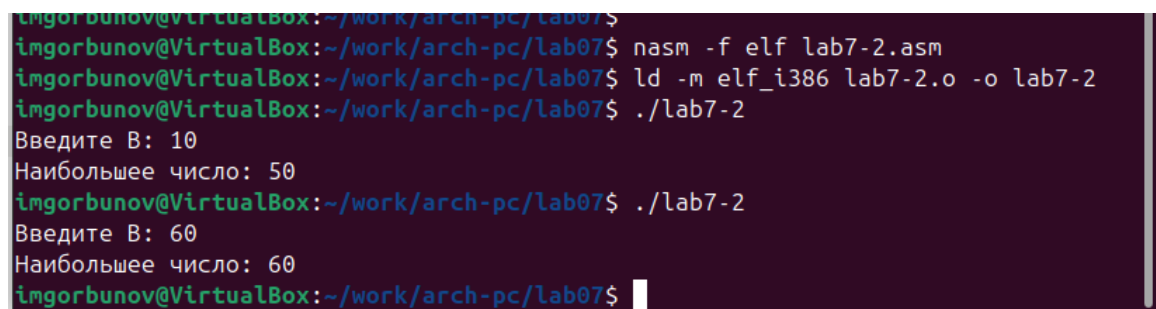
3. Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую 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.



```
1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',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 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]
```

Рисунок 2.7: Программа в файле lab7-2.asm

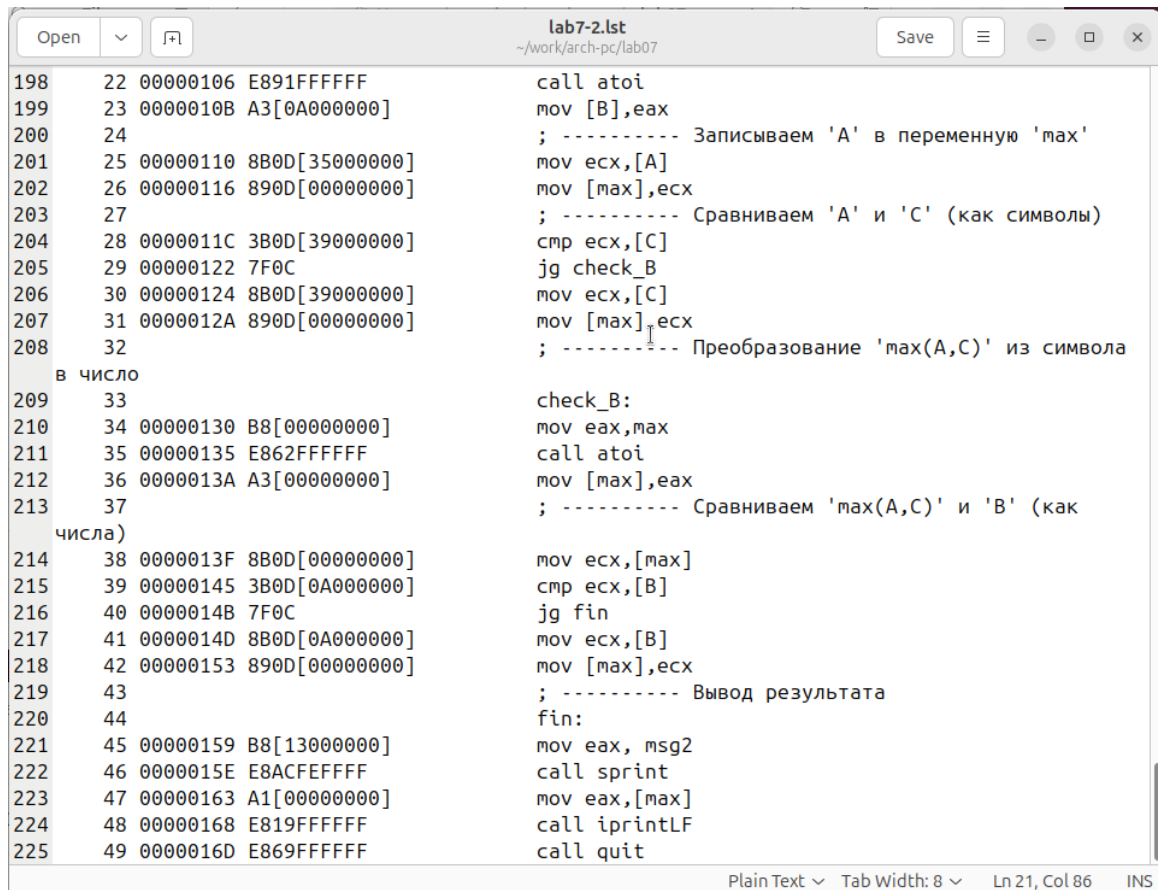


```
imgorbunov@VirtualBox:~/work/arch-pc/lab07$
imgorbunov@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
imgorbunov@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
imgorbunov@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
imgorbunov@VirtualBox:~/work/arch-pc/lab07$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

Рисунок 2.8: Запуск программы lab7-2.asm

4. 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`



```
198 22 00000106 E891FFFFFF call atoi
199 23 0000010B A3[0A000000] mov [B],eax
200 24 ; ----- Записываем 'A' в переменную 'max'
201 25 00000110 8B0D[35000000] mov ecx,[A]
202 26 00000116 890D[00000000] mov [max],ecx
203 27 ; ----- Сравниваем 'A' и 'C' (как символы)
204 28 0000011C 3B0D[39000000] cmp ecx,[C]
205 29 00000122 7F0C jg check_B
206 30 00000124 8B0D[39000000] mov ecx,[C]
207 31 0000012A 890D[00000000] mov [max],ecx
208 32 ; ----- Преобразование 'max(A,C)' из символа
    в число
209 33
210 34 00000130 B8[00000000] check_B:
211 35 00000135 E862FFFFFF mov eax,max
212 36 0000013A A3[00000000] call atoi
213 37 mov [max],eax
214 38 0000013F 8B0D[00000000] ; ----- Сравниваем 'max(A,C)' и 'B' (как
    числа)
215 39 00000145 3B0D[0A000000] mov ecx,[max]
216 40 0000014B 7F0C cmp ecx,[B]
217 41 0000014D 8B0D[0A000000] jg fin
218 42 00000153 890D[00000000] mov ecx,[B]
219 43 mov [max],ecx
220 44 ; ----- Вывод результата
221 45 00000159 B8[13000000] fin:
222 46 0000015E E8ACFEFFFF mov eax, msg2
223 47 00000163 A1[00000000] call sprint
224 48 00000168 E819FFFFFF mov eax,[max]
225 49 0000016D E869FFFFFF call iprintLF
226 50 00000172 00000000 call quit
```

Рисунок 2.9: Файл листинга `lab7-2`

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

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

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

```
imgorbunov@VirtualBox:~/work/arch-pc/lab07$
imgorbunov@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
imgorbunov@VirtualBox:~/work/arch-pc/lab07$
imgorbunov@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:18: error: invalid combination of opcode and operands
imgorbunov@VirtualBox:~/work/arch-pc/lab07$
```

Рисунок 2.10: Ошибка трансляции lab7-2

```

lab7-2.lst
~/work/arch-pc/lab07
lab7-2.asm
lab7-2.lst
187 11 global _start
188 12 _start:
189 13 ; ----- Вывод сообщения 'Введите B: '
190 14 000000E8 B8[00000000] mov eax,msg1
191 15 000000ED E81DFFFFFF call sprint
192 16 ; ----- Ввод 'B'
193 17 000000F2 B9[0A000000] mov ecx,B
194 18 mov edx,
195 18 ***** error: invalid combination of opcode and operands
196 19 000000F7 E847FFFFFF call sread
197 20 ; ----- Преобразование 'B' из символа в число
198 21 000000FC B8[0A000000] mov eax,B
199 22 00000101 E896FFFFFF call atoi
200 23 00000106 A3[0A000000] mov [B],eax
201 24 ; ----- Записываем 'A' в переменную 'max'
202 25 0000010B 8B0D[35000000] mov ecx,[A]
203 26 00000111 890D[00000000] mov [max],ecx
204 27 ; ----- Сравниваем 'A' и 'C' (как символы)
205 28 00000117 3B0D[39000000] cmp ecx,[C]
206 29 0000011D 7F0C jg check_B
207 30 0000011F 8B0D[39000000] mov ecx,[C]
208 31 00000125 890D[00000000] mov [max],ecx
209 32 ; ----- Преобразование 'max(A,C)' из символа
в число
210 33 check_B:
211 34 0000012B B8[00000000] mov eax,max
212 35 00000130 E867FFFFFF call atoi
213 36 00000135 A3[00000000] mov [max],eax
214 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как

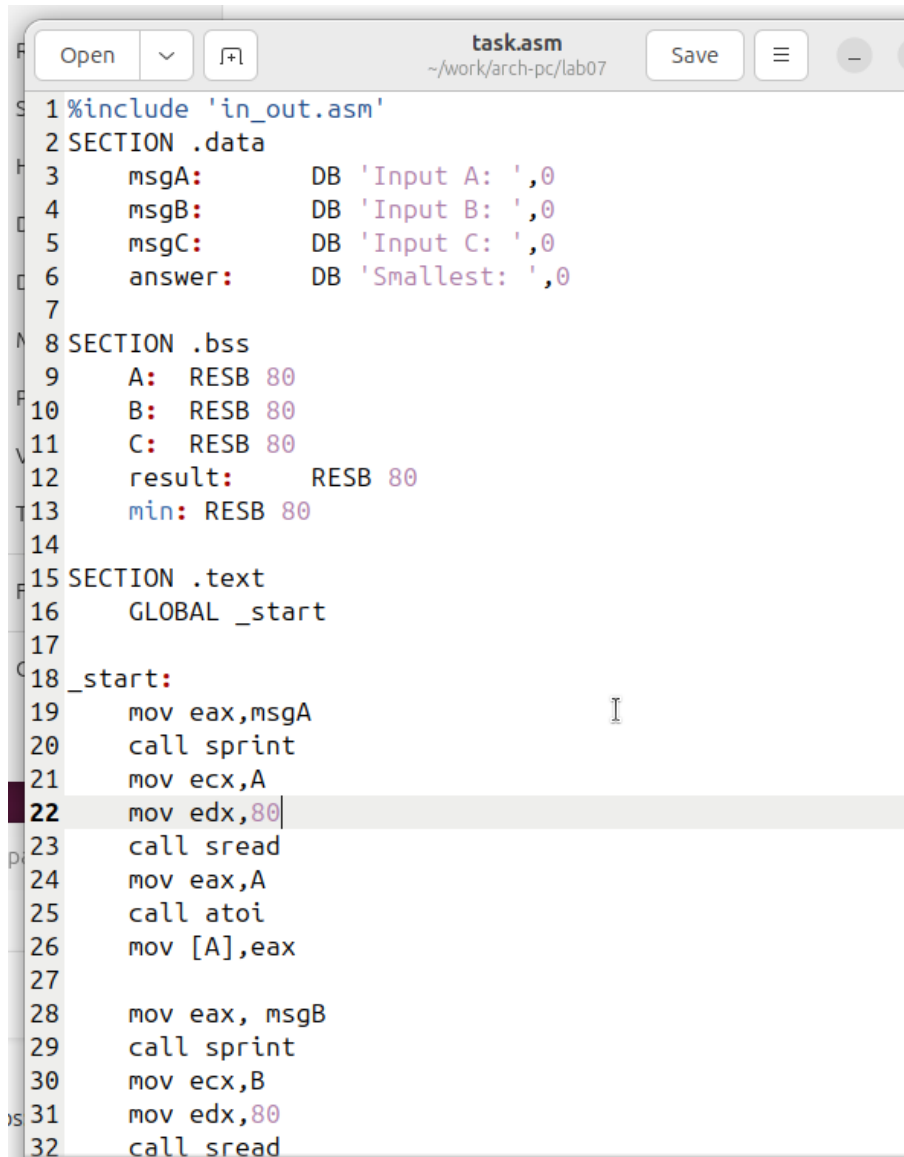
```

Рисунок 2.11: Файл листинга с ошибкой lab7-2

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

5. 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 3 - 45,67,15



```
1 %include 'in_out.asm'
2 SECTION .data
3     msgA:      DB 'Input A: ',0
4     msgB:      DB 'Input B: ',0
5     msgC:      DB 'Input C: ',0
6     answer:    DB 'Smallest: ',0
7
8 SECTION .bss
9     A:  RESB 80
10    B:  RESB 80
11    C:  RESB 80
12    result:  RESB 80
13    min: RESB 80
14
15 SECTION .text
16     GLOBAL _start
17
18 _start:
19     mov eax,msgA
20     call sprint
21     mov ecx,A
22     mov edx,80
23     call sread
24     mov eax,A
25     call atoi
26     mov [A],eax
27
28     mov eax, msgB
29     call sprint
30     mov ecx,B
31     mov edx,80
32     call sread
```

Рисунок 2.12: Программа в файле task.asm

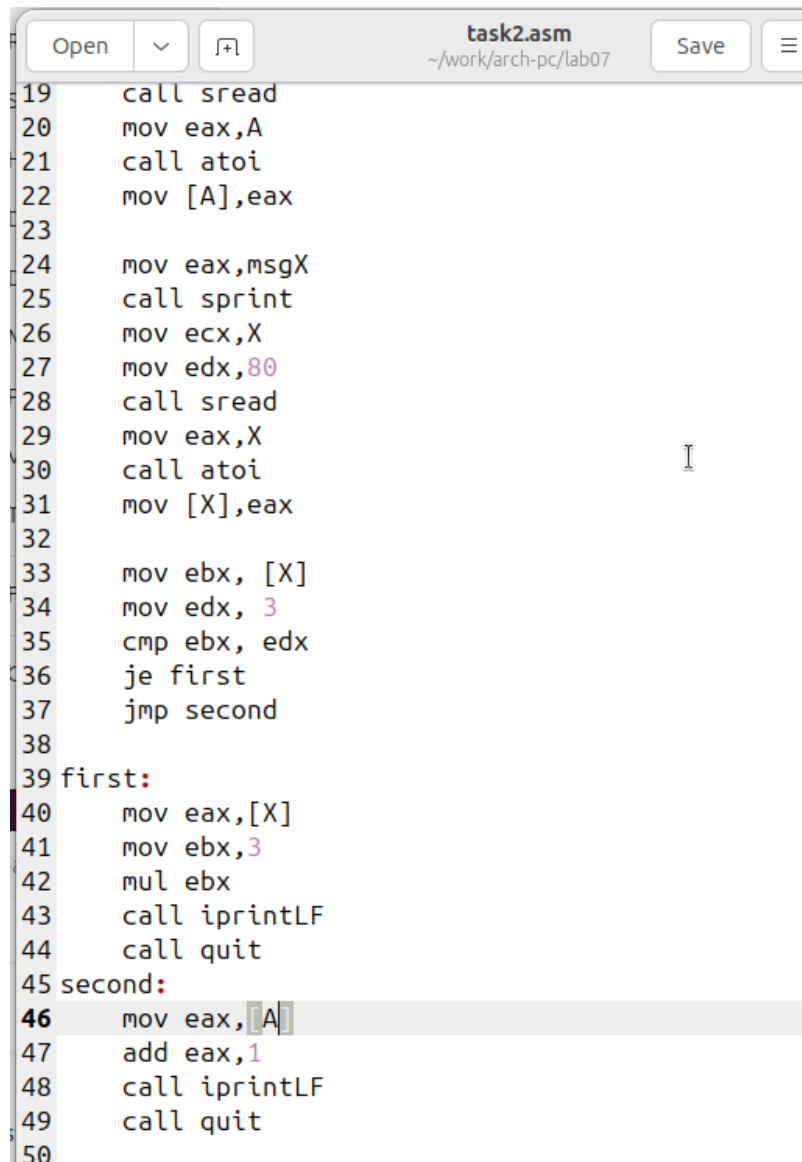

```
imgorbunov@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf task.asm
imgorbunov@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 task.o -o task.o
ld: input file 'task.o' is the same as output file
imgorbunov@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 task.o -o task
imgorbunov@VirtualBox:~/work/arch-pc/lab07$ ./task
Input A: 45
Input B: 67
Input C: 15
Smallest: 15
imgorbunov@VirtualBox:~/work/arch-pc/lab07$
```

Рисунок 2.13: Запуск программы task.asm

6. 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 3

$$\begin{cases} 3x, x = 3 \\ a + 1, x \neq 3 \end{cases}$$



```
task2.asm
~/work/arch-pc/lab07
Open  Save

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, 3
35  cmp  ebx, edx
36  je  first
37  jmp  second
38
39 first:
40  mov  eax,[X]
41  mov  ebx,3
42  mul  ebx
43  call iprintLF
44  call quit
45 second:
46  mov  eax,[A]
47  add  eax,1
48  call iprintLF
49  call quit
50
```

Рисунок 2.14: Программа в файле task2.asm

```
imgorbunov@VirtualBox:~/work/arch-pc/lab07$  
imgorbunov@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf task2.asm  
imgorbunov@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 task2.o -o task2  
imgorbunov@VirtualBox:~/work/arch-pc/lab07$ ./task2  
Input A: 4  
Input X: 3  
9  
imgorbunov@VirtualBox:~/work/arch-pc/lab07$ ./task2  
Input A: 4  
Input X: 1  
5  
imgorbunov@VirtualBox:~/work/arch-pc/lab07$
```

Рисунок 2.15: Запуск программы task2.asm

3 Выводы

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