

Отчёт по лабораторной работе 7

Архитектура компьютера

Эргешов Байрам НКАбд-02-23

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Задание для самостоятельной работы	16
3	Выводы	21

Список иллюстраций

2.1	Код программы lab7-1.asm	7
2.2	Компиляция и запуск программы lab7-1.asm	8
2.3	Код программы lab7-1.asm	9
2.4	Компиляция и запуск программы lab7-1.asm	10
2.5	Код программы lab7-1.asm	11
2.6	Компиляция и запуск программы lab7-1.asm	12
2.7	Код программы lab7-2.asm	13
2.8	Компиляция и запуск программы lab7-2.asm	13
2.9	Файл листинга lab7-2	14
2.10	Ошибка трансляции lab7-2	15
2.11	Файл листинга с ошибкой lab7-2	16
2.12	Код программы program-1.asm	17
2.13	Компиляция и запуск программы program-1.asm	18
2.14	Код программы program-2.asm	19
2.15	Компиляция и запуск программы program-2.asm	20

Список таблиц

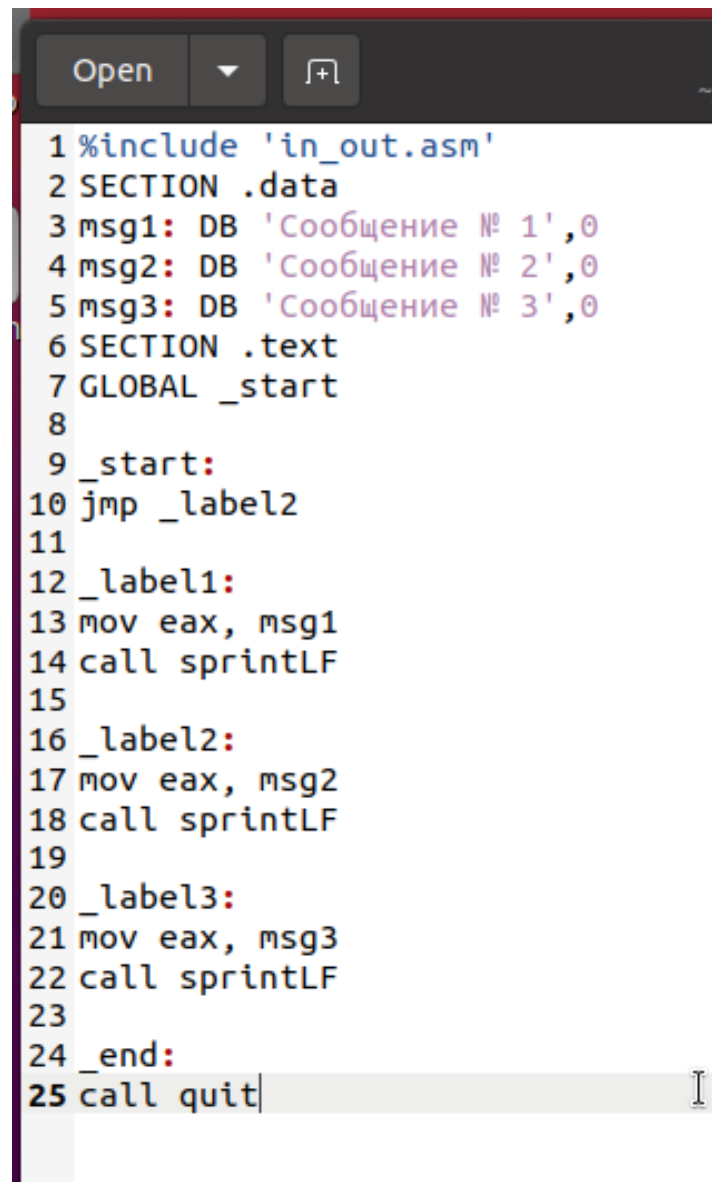
1 Цель работы

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

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

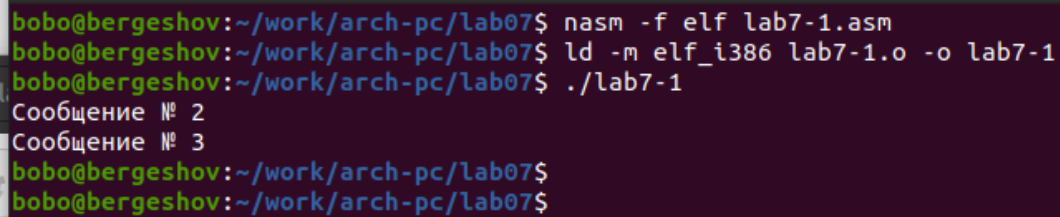
Инструкция jmp в NASM используется для выполнения безусловных переходов. Я рассмотрел пример программы, в которой использовалась инструкция jmp. Написал текст программы из листинга 7.1 в файле lab7-1.asm.



```
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 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.

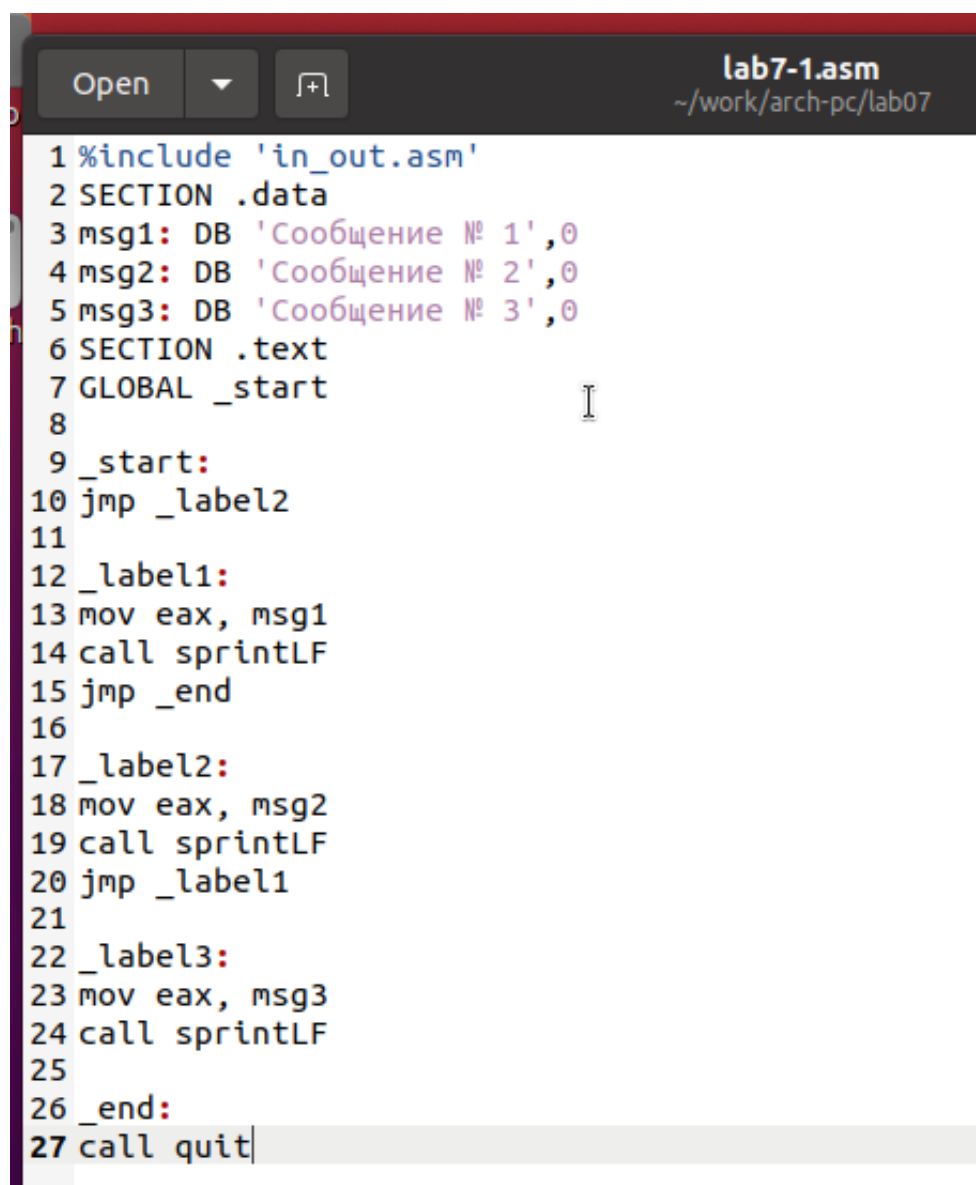
A terminal window with a dark purple background and light green text. The prompt is 'bobo@bergeshov:~/work/arch-pc/lab07\$'. The user enters 'nasm -f elf lab7-1.asm', followed by 'ld -m elf_i386 lab7-1.o -o lab7-1', and then './lab7-1'. The output shows 'Сообщение № 2' and 'Сообщение № 3' on separate lines. The prompt then changes to 'bobo@bergeshov:~/work/arch-pc/lab07\$' again.

```
bobo@bergeshov:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
bobo@bergeshov:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
bobo@bergeshov:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
bobo@bergeshov:~/work/arch-pc/lab07$
bobo@bergeshov:~/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. Я изменил программу так, чтобы она сначала выводила “Сообщение № 2”, затем “Сообщение № 1” и завершала работу. Для этого я добавил в текст программы после вывода “Сообщения № 2” инструкцию `jmp` с меткой `_label1` (чтобы перейти к инструкциям вывода “Сообщения № 1”) и после вывода “Сообщения № 1” добавил инструкцию `jmp` с меткой `_end` (чтобы перейти 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
```

Рис. 2.3: Код программы lab7-1.asm

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

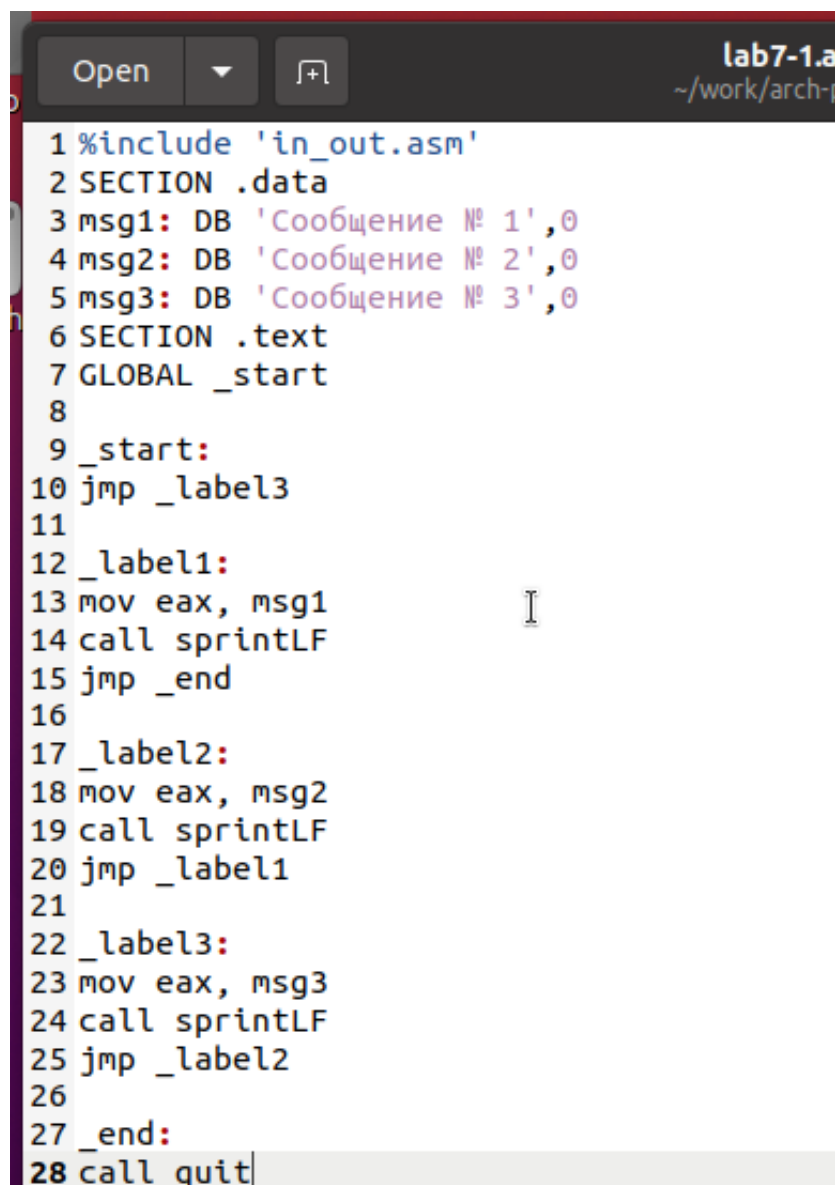
Рис. 2.4: Компиляция и запуск программы lab7-1.asm

Я 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: Код программы lab7-1.asm

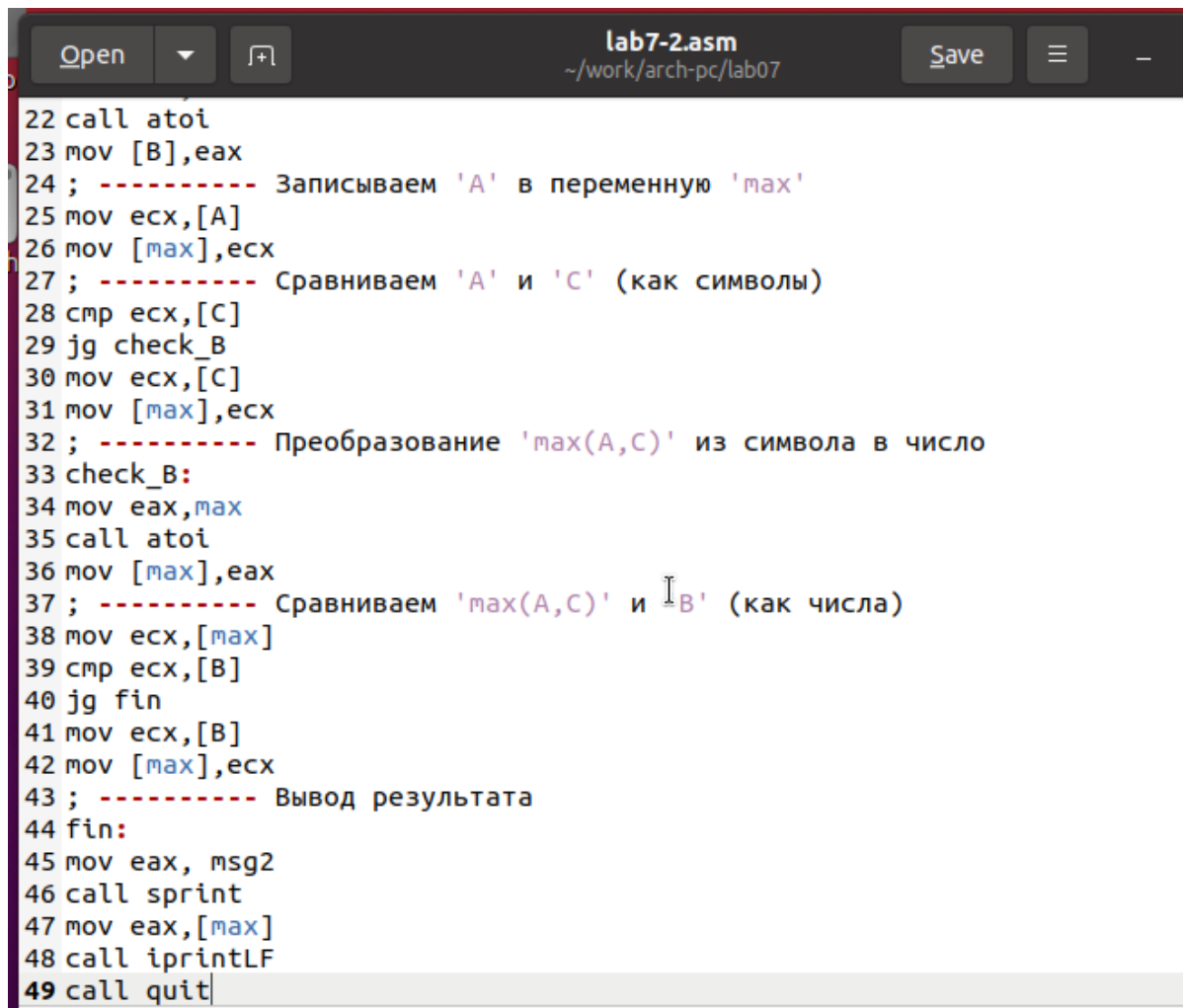
```
bobo@bergeshov:~/work/arch-pc/lab07$  
bobo@bergeshov:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm  
bobo@bergeshov:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1  
bobo@bergeshov:~/work/arch-pc/lab07$ ./lab7-1  
Сообщение № 3  
Сообщение № 2  
Сообщение № 1  
bobo@bergeshov:~/work/arch-pc/lab07$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

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.

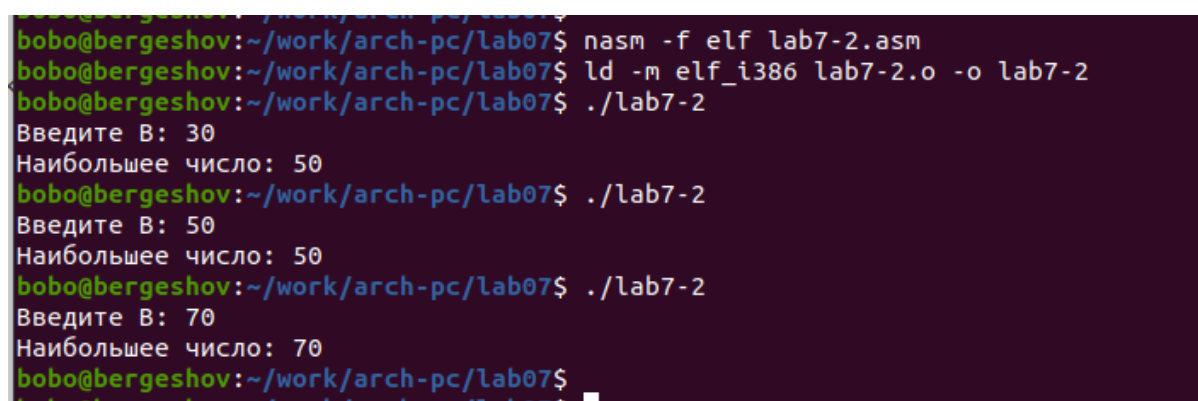
Создал исполняемый файл и проверил его работу для разных значений В.



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

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



```
bobo@bergeshov:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
bobo@bergeshov:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
bobo@bergeshov:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 30
Наибольшее число: 50
bobo@bergeshov:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 50
Наибольшее число: 50
bobo@bergeshov:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 70
Наибольшее число: 70
bobo@bergeshov:~/work/arch-pc/lab07$
```

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

Обычно `nasm` создаёт в результате ассемблирования только объектный файл. Чтобы получить файл листинга, нужно указать ключ `-l` и задать имя файла листинга в командной строке.

Создал файл листинга для программы из файла `lab7-2.asm`

lab7-2.asm			lab7-2.lst
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 E8ACFEFFFF	call sprintf
222	47	00000163 A1[00000000]	mov eax,[max]
223	48	00000168 E819FFFFFF	call iprintLF
224	49	0000016D E869FFFFFF	call quit

Рис. 2.9: Файл листинга lab7-2

Я внимательно ознакомился с форматом и содержимым файла листинга. Подробно объясню содержимое трёх строк из этого файла.

строка 213

- 38 - номер строки в подпрограмме
- 0000013F - адрес
- 8B0D[00000000] - машинный код
- `mov ecx,[max]` - код программы - копирует MAX в ecx

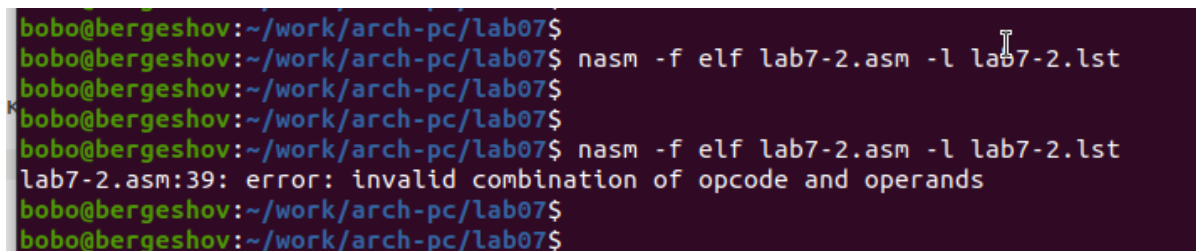
строка 214

- 39 - номер строки в подпрограмме
- 00000145 - адрес
- 3B0D[0A000000] - машинный код
- `cmp esx,[B]` - код программы - сравнивает `esx` и `B`

строка 215

- 40 - номер строки в подпрограмме
- 0000014B - адрес
- 7F0C - машинный код
- `jb fin` - код программы - если больше перейти к метке `fin`

Далее я открыл файл с программой `lab7-2.asm` и в инструкции с двумя операндами удалил один из операндов. Затем выполнил трансляцию с получением файла листинга.



```

bobo@bergeshov:~/work/arch-pc/lab07$
bobo@bergeshov:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
bobo@bergeshov:~/work/arch-pc/lab07$
bobo@bergeshov:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:39: error: invalid combination of opcode and operands
bobo@bergeshov:~/work/arch-pc/lab07$
bobo@bergeshov:~/work/arch-pc/lab07$

```

Рис. 2.10: Ошибка трансляции `lab7-2`

```

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 cmp ecx,[B]
215 39 ***** error: invalid combination of opcode and operands
216 40 00000145 7F0C jg fin
217 41 00000147 8B0D[0A000000] mov ecx,[B]
218 42 0000014D 890D[00000000] mov [max],ecx
219 43 ; ----- Вывод результата
220 44 fin:
221 45 00000153 B8[13000000] mov eax,msg2
222 46 00000158 E8B2FFFFFF call sprint
223 47 0000015D A1[00000000] mov eax,[max]
224 48 00000162 E81FFFFFFF call iprintLF
225 49 00000167 F86FFFFFFF call quit

```

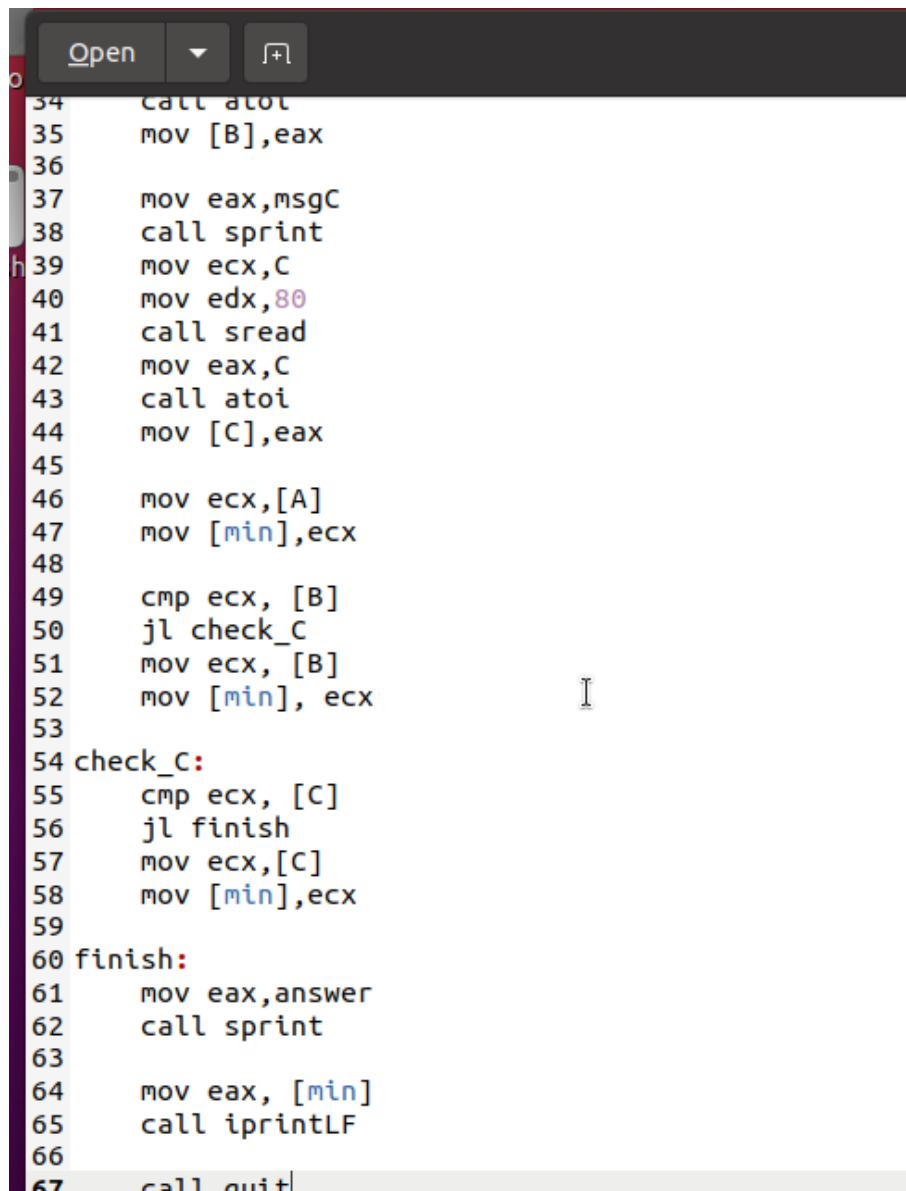
Рис. 2.11: Файл листинга с ошибкой lab7-2

В результате объектный файл не смог быть создан из-за ошибки. Но был получен файл листинга, в котором выделено место ошибки.

2.1 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

Мой вариант 1 - числа: 17,23,45



```
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
```

Рис. 2.12: Код программы program-1.asm

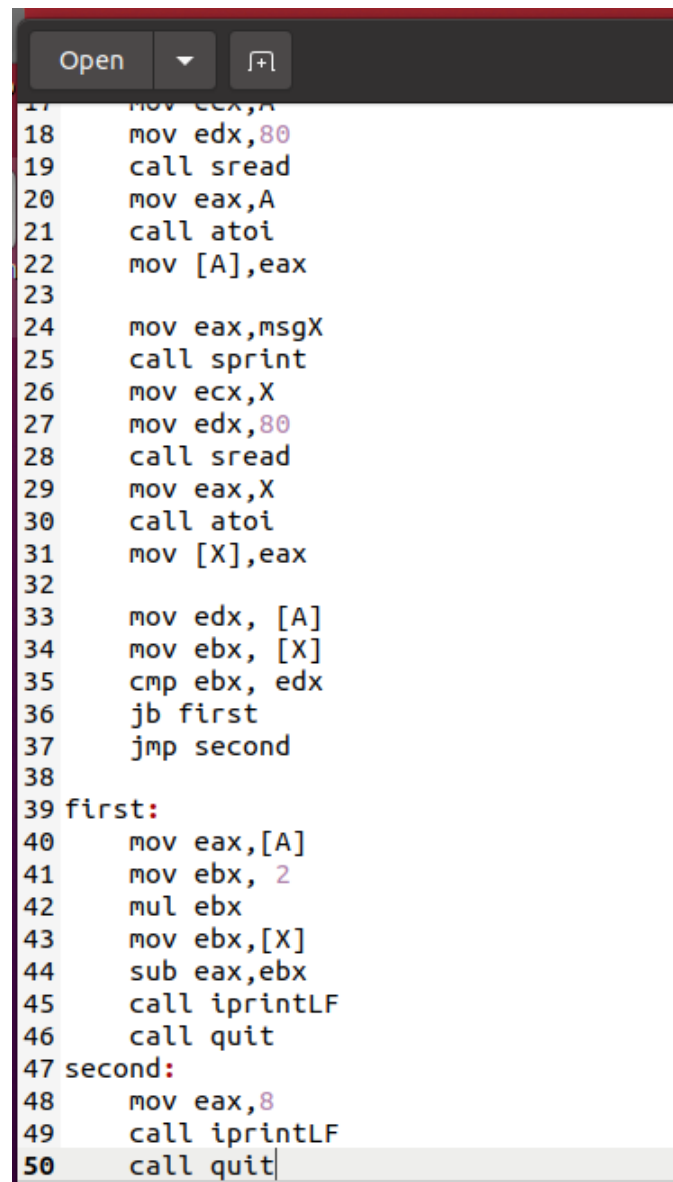
```
bobo@bergeshov:~/work/arch-pc/lab07$  
bobo@bergeshov:~/work/arch-pc/lab07$ nasm -f elf program-1.asm  
bobo@bergeshov:~/work/arch-pc/lab07$ ld -m elf_i386 program-1.o -o program-1  
bobo@bergeshov:~/work/arch-pc/lab07$ ./program-1  
Input A: 17  
Input B: 23  
Input C: 45  
Smallest: 17  
bobo@bergeshov:~/work/arch-pc/lab07$  
bobo@bergeshov:~/work/arch-pc/lab07$
```

Рис. 2.13: Компиляция и запуск программы program-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.

Мой вариант 1

$$\begin{cases} 2a - x, x < a \\ 8, x \geq a \end{cases}$$



```

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 sprintf
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, [A]
34  mov ebx, [X]
35  cmp ebx, edx
36  jb first
37  jmp second
38
39 first:
40  mov eax,[A]
41  mov ebx, 2
42  mul ebx
43  mov ebx,[X]
44  sub eax,ebx
45  call iprintLF
46  call quit
47 second:
48  mov eax,8
49  call iprintLF
50  call quit

```

Рис. 2.14: Код программы program-2.asm

При $x = 1, a = 2f(x) = 3$

При $x = 2, a = 1f(x) = 8$

```
bobo@bergeshov:~/work/arch-pc/lab07$  
bobo@bergeshov:~/work/arch-pc/lab07$ nasm -f elf program-2.asm  
bobo@bergeshov:~/work/arch-pc/lab07$ ld -m elf_i386 program-2.o -o program-2  
bobo@bergeshov:~/work/arch-pc/lab07$ ./program-2  
Input A: 2  
Input X: 1  
3  
bobo@bergeshov:~/work/arch-pc/lab07$ ./program-2  
Input A: 1  
Input X: 2  
8  
bobo@bergeshov:~/work/arch-pc/lab07$
```

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

3 Выводы

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