

Отчёт по лабораторной работе 7

Архитектура компьютера

Амарбаяр Чинхусэл

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	18

Список иллюстраций

2.1	Программа lab7-1.asm	7
2.2	Запуск программы lab7-1.asm	8
2.3	Программа lab7-1.asm	9
2.4	Запуск программы lab7-1.asm	9
2.5	Программа lab7-1.asm	10
2.6	Запуск программы lab7-1.asm	11
2.7	Программа 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 prog1.asm	16
2.13	Запуск программы prog1.asm	17

Список таблиц

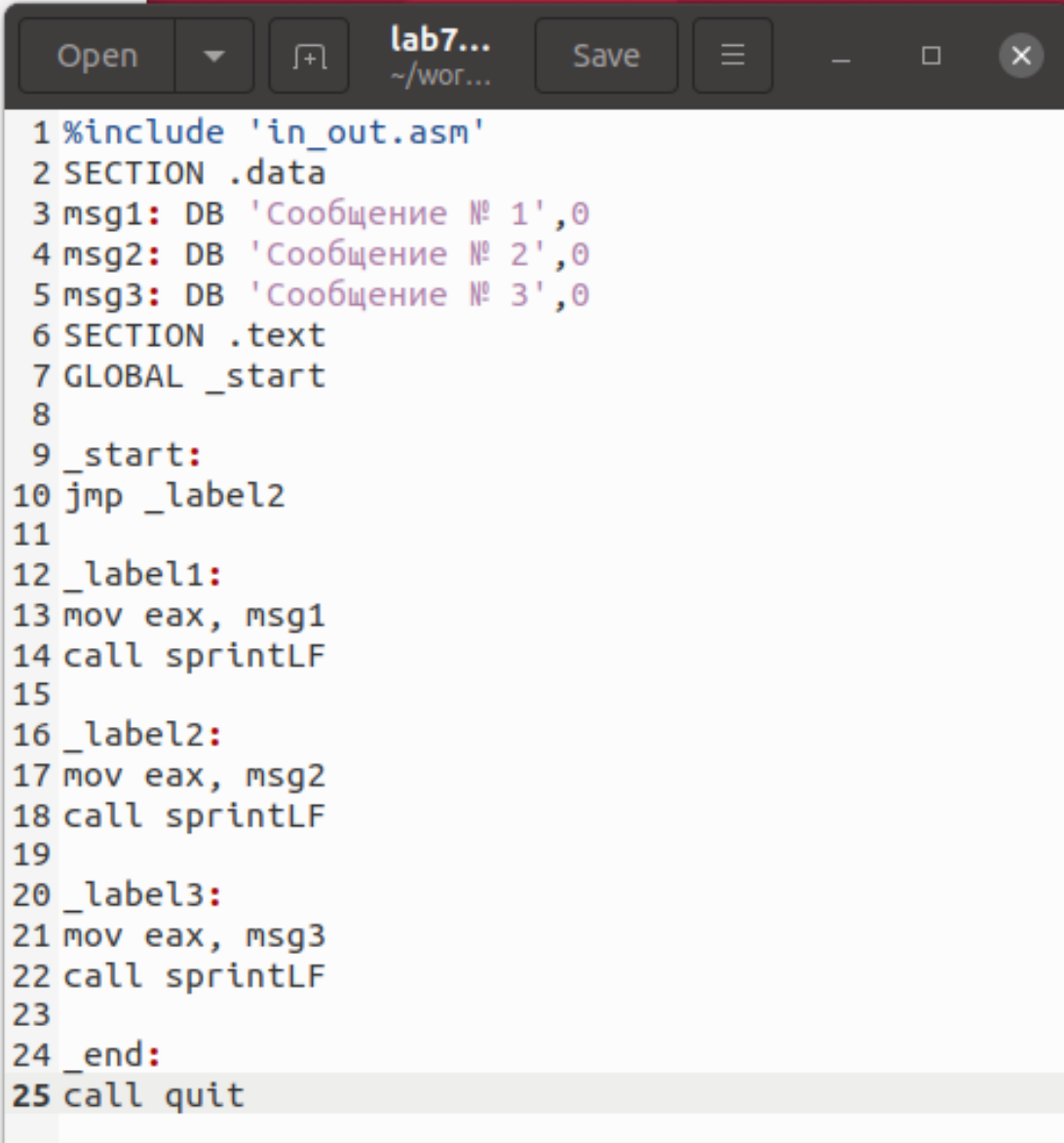
1 Цель работы

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

1. Создал каталог для программам 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`.

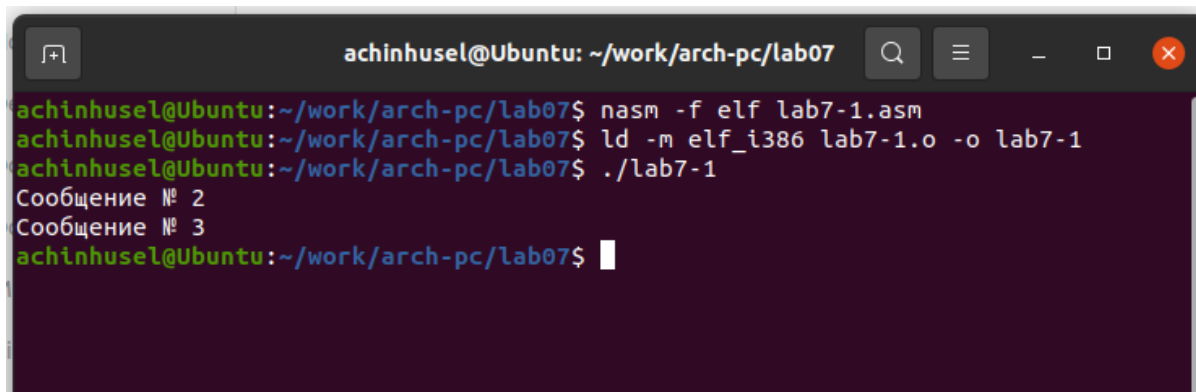
Написал в файл lab7-1.asm текст программы из листинга 7.1.

A screenshot of a text editor window with a dark theme. The window title bar shows 'lab7...' and a file path '~/.wof...'. The editor contains assembly code for a program named lab7-1.asm. The code includes three data messages in Russian, three labels for printing them, and a main loop that prints each message and then jumps to the next label. The code ends with a quit call.

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

Рис. 2.1: Программа lab7-1.asm

Создал исполняемый файл и запустил его.

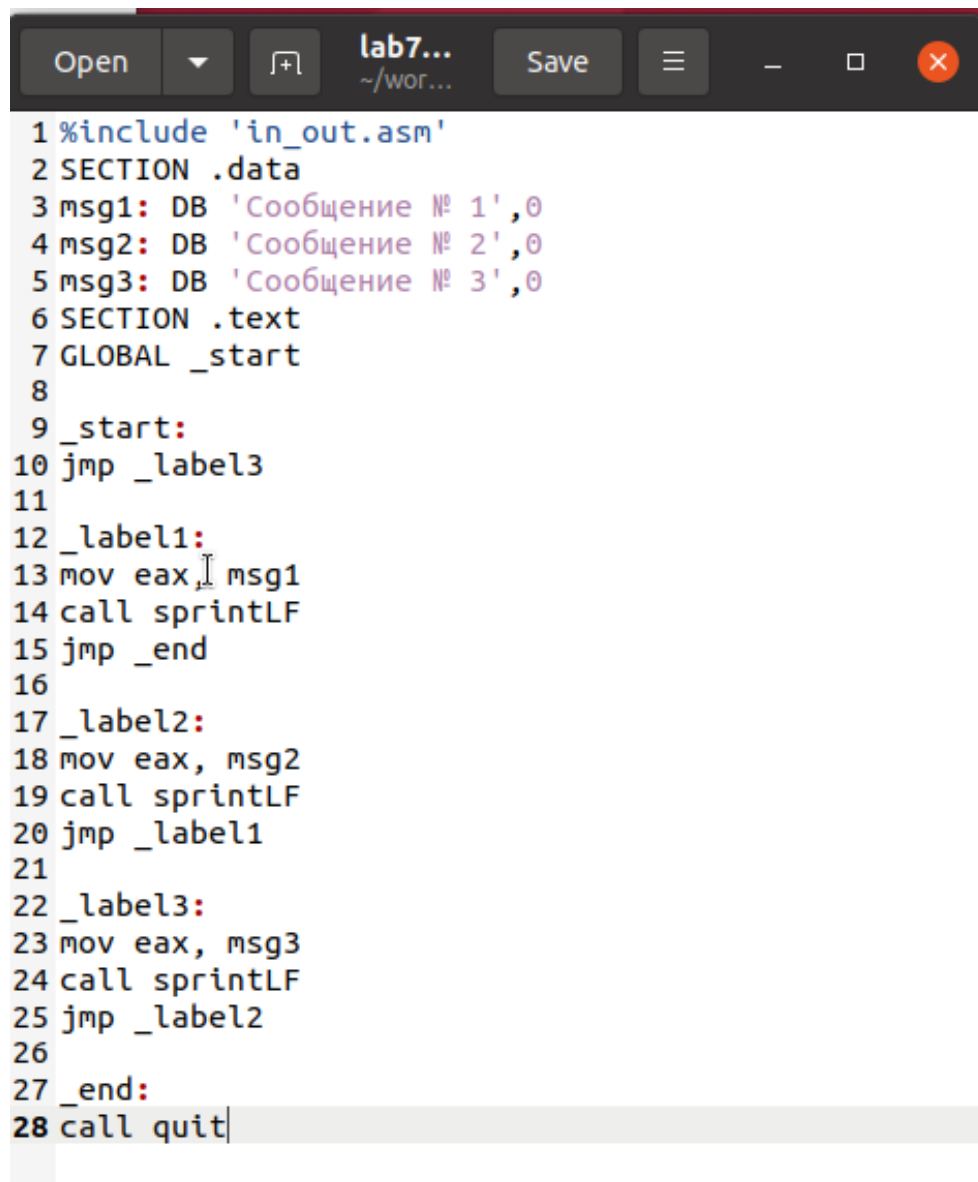
A terminal window titled 'achinhusel@Ubuntu: ~/work/arch-pc/lab07'. The prompt is 'achinhusel@Ubuntu:~/work/arch-pc/lab07\$'. The user enters 'nasm -f elf lab7-1.asm', followed by 'ld -m elf_i386 lab7-1.o -o lab7-1', and finally './lab7-1'. The output shows 'Сообщение № 2' and 'Сообщение № 3' on separate lines. The prompt returns after the execution.

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

Рис. 2.2: Запуск программы lab7-1.asm

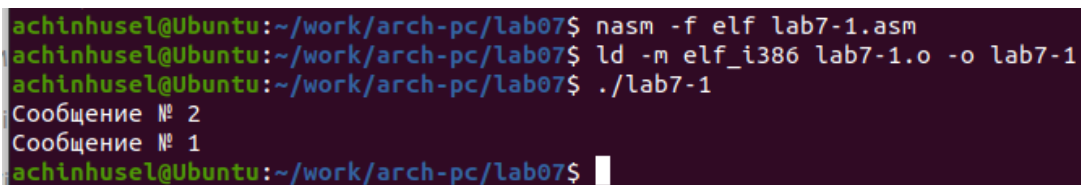
Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е.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

A screenshot of a text editor window titled 'lab7...' with a dark theme. The window contains assembly code for a program. The code starts with including 'in_out.asm', defining three data messages in Russian, and then a text section with labels _start, _label1, _label2, _label3, and _end. The logic prints the messages in reverse order: first _label3, then _label2, then _label1, and finally _end (which calls quit).

```
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. 2.3: Программа lab7-1.asm

A screenshot of a terminal window showing the compilation and execution of the assembly program. The user runs 'nasm -f elf lab7-1.asm', then 'ld -m elf_i386 lab7-1.o -o lab7-1', and finally './lab7-1'. The output shows the messages being printed in reverse order: 'Сообщение № 2', 'Сообщение № 1', and then the prompt returns.

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

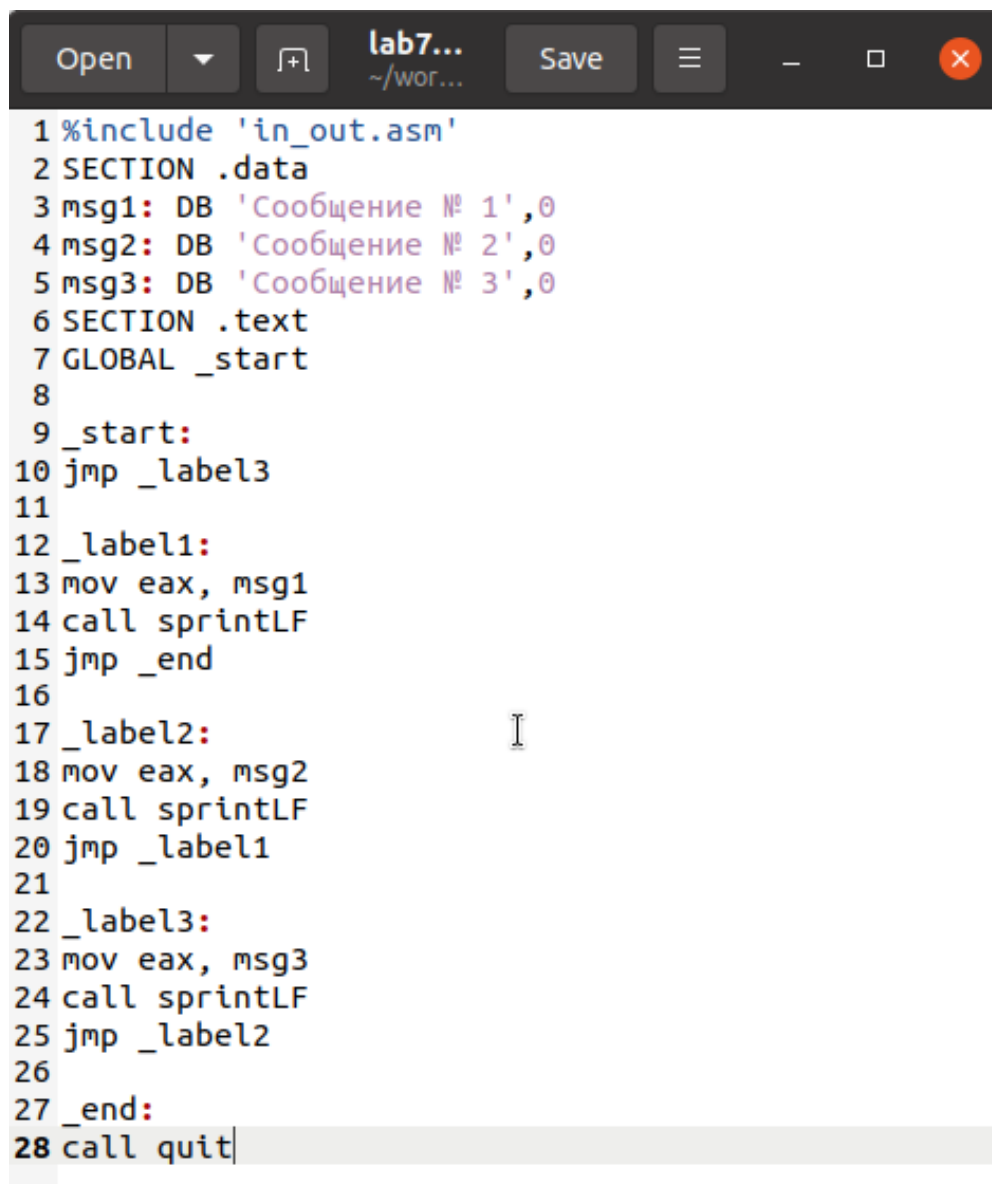
Рис. 2.4: Запуск программы lab7-1.asm

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
```

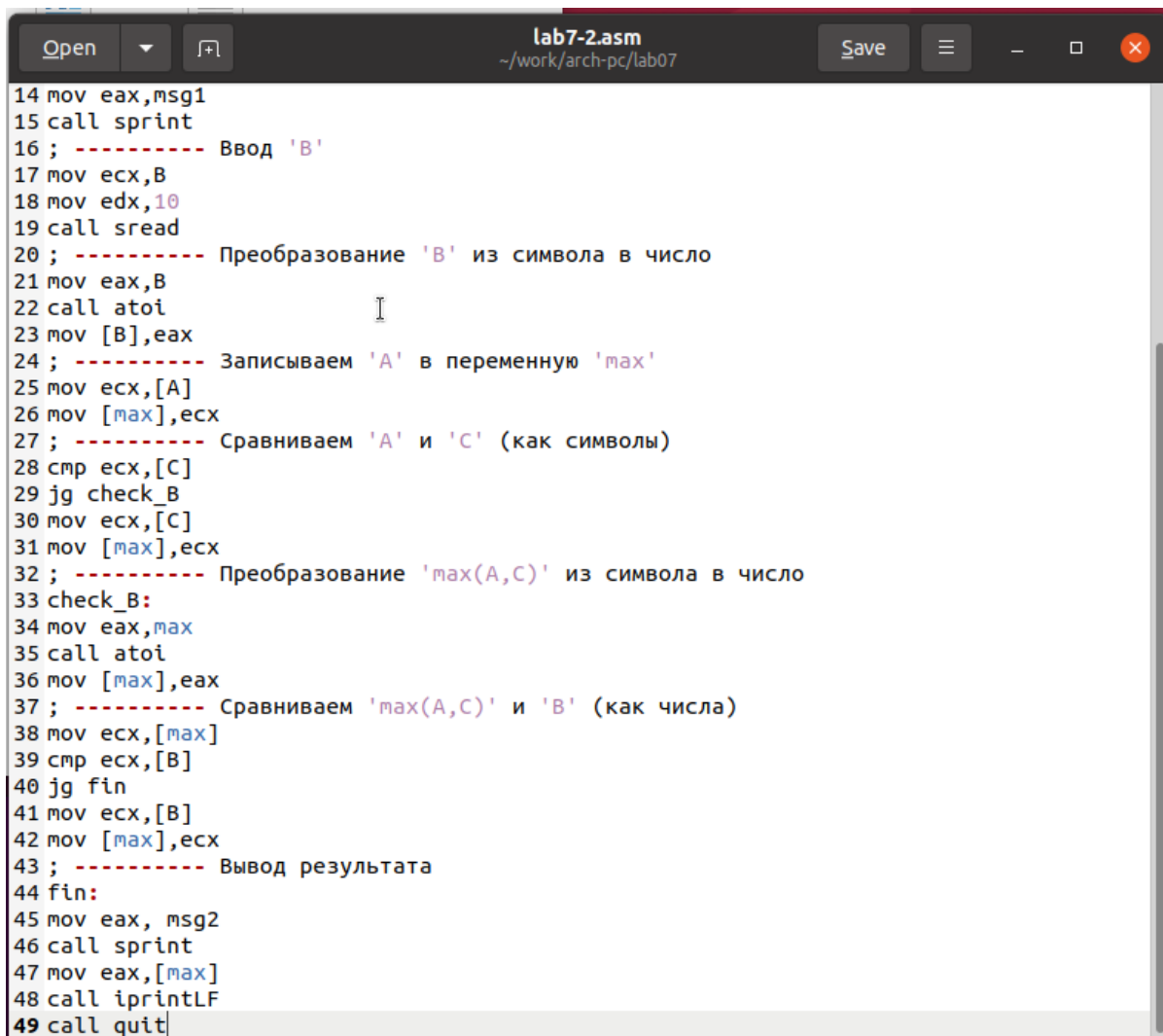
Рис. 2.5: Программа lab7-1.asm

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

Рис. 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.

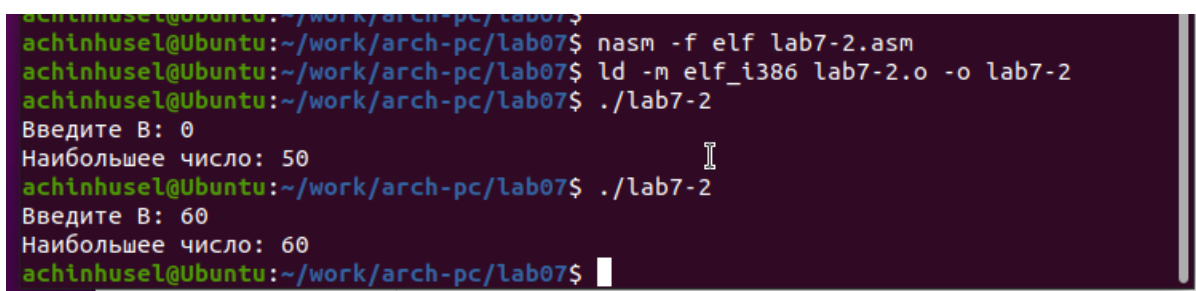
Создал исполняемый файл и проверил его работу для разных значений B.



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

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 lab7-2.asm



```
achinhuse1@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
achinhuse1@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
achinhuse1@Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 0
Наибольшее число: 50
achinhuse1@Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
achinhuse1@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 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`

```
lab7-2.lst
~/work/arch-pc/lab07

lab7-2.asm
lab7-2.lst

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 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
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 E8ACFEFFFF mov eax,msg2
222 47 00000163 A1[00000000] call sprint
223 48 00000168 E819FFFFFF mov eax,[max]
224 49 0000016D E869FFFFFF call iprintLF
call quit
```

Рис. 2.9: Файл листинга lab7-2

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

*строка 209

-34 - номер строки

- 00000130 - адрес

- B8[00000000] - машинный код
- mov eax, max - код программы
- *строка 210
- 35 - номер строки
- 00000135 - адрес
- E862FFFFFF - машинный код
- call atoi - код программы
- *строка 211
- 36 - номер строки
- 0000013A - 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.

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

Рис. 2.10: Ошибка трансляции lab7-2

```

lab7-2.asm
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 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 sprintf
223 47 0000015E A1[00000000] mov eax,[max]
224 48 00000163 E81EFFFFFF call iprintf
225 49 00000168 E86EFFFFFF call quit

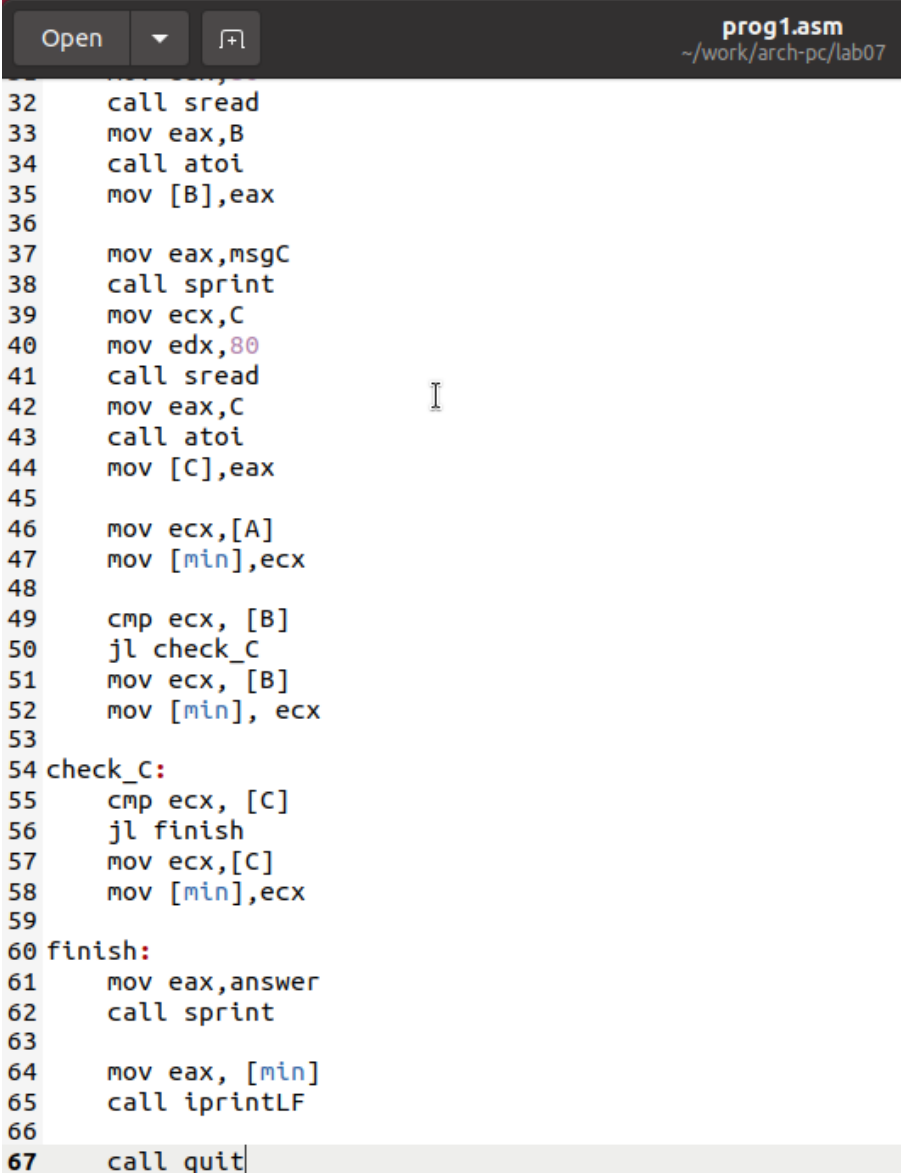
```

Рис. 2.11: Файл листинга с ошибкой lab7-2

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

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 20 - 92, 2, 61



```
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
```

Рис. 2.12: Программа prog1.asm


```
achinhusel@Ubuntu:~/work/arch-pc/lab07$  
achinhusel@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf prog1.asm  
achinhusel@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 prog1.o -o prog1  
achinhusel@Ubuntu:~/work/arch-pc/lab07$ ./prog1  
Input A: 92  
Input B: 2  
Input C: 61  
Smallest: 2  
achinhusel@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.13: Запуск программы prog1.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 20

$$\begin{cases} x - a, x \geq a \\ 5, x < a \end{cases}$$

[Программа prog2.asm](image/14.png){ #fig:014 width=70%, height=70% }

Подставляем $x = 1, a = 2$ и $x = 2, a = 1$

[Запуск программы prog2.asm](image/15.png){ #fig:015 width=70%, height=70%

}

3 Выводы

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