

# **Отчёта по лабораторной работе 7**

**Команды безусловного и условного переходов в Nasm.  
Программирование ветвлений**

Гайбуллаев Фаррух Шухрат

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	28

## Список иллюстраций

2.1	Файл lab7-1.asm . . . . .	7
2.2	Запуск программы lab7-1.asm . . . . .	8
2.3	Файл lab7-1.asm: . . . . .	9
2.4	Запуск программы lab7-1.asm: . . . . .	11
2.5	Файл lab7-1.asm . . . . .	12
2.6	Запуск программы lab7-1.asm . . . . .	13
2.7	Файл lab7-2.asm . . . . .	15
2.8	Запуск программы lab7-2.asm . . . . .	17
2.9	Файл листинга lab7-2 . . . . .	18
2.10	ошибка трансляции lab7-2 . . . . .	19
2.11	файл листинга с ошибкой lab7-2 . . . . .	20
2.12	Файл lab7-3.asm . . . . .	21
2.13	Запуск программы lab7-3.asm . . . . .	24
2.14	Файл lab7-4.asm . . . . .	25
2.15	Запуск программы lab7-4.asm . . . . .	27

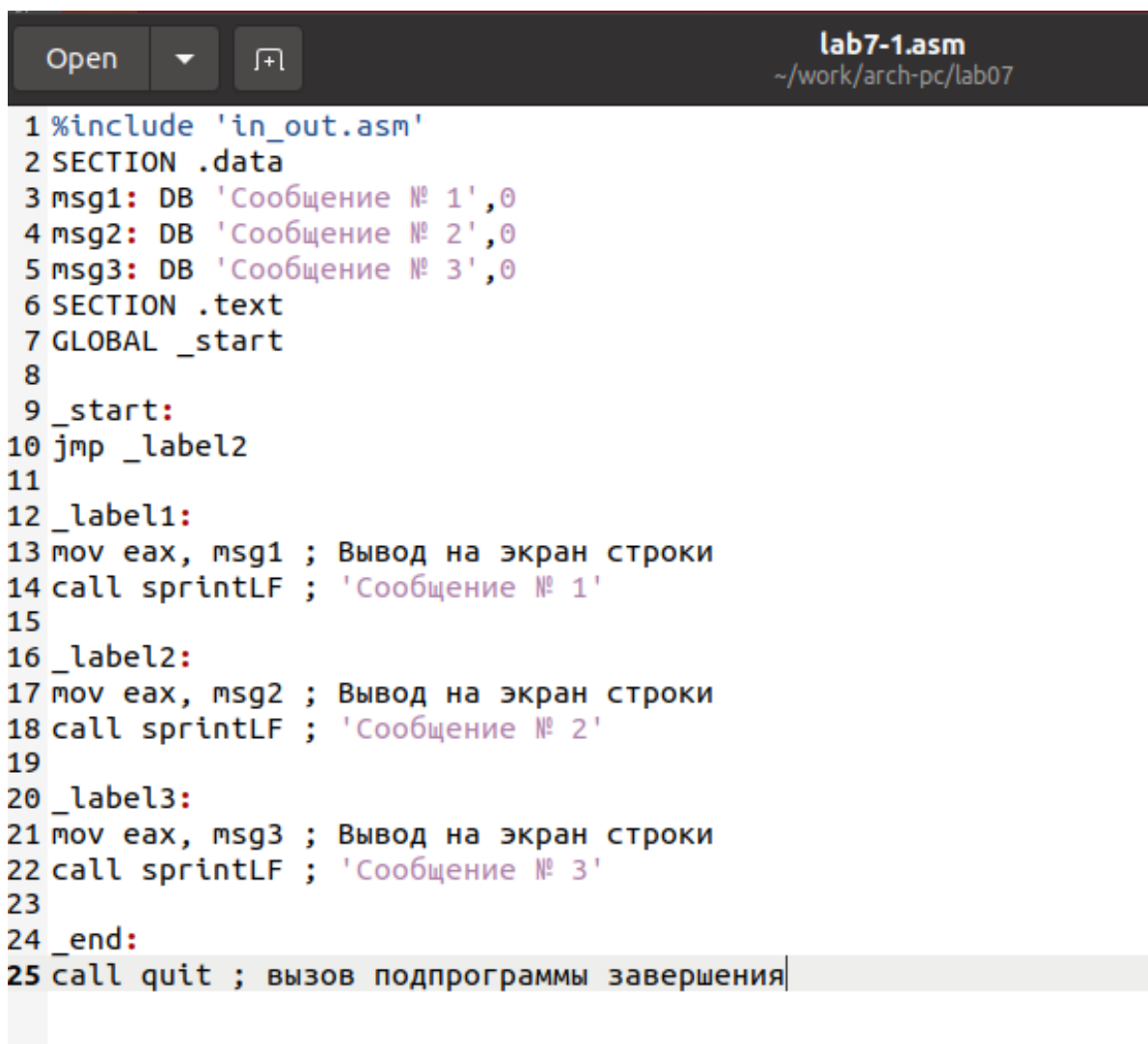
## Список таблиц

# 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. Создайте каталог для программ лабораторной работы № 7, перейдите в него и создайте файл lab7-1.asm
2. Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Введите 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 ; Вывод на экран строки
14 call sprintf ; 'Сообщение № 1'
15
16 _label2:
17 mov eax, msg2 ; Вывод на экран строки
18 call sprintf ; 'Сообщение № 2'
19
20 _label3:
21 mov eax, msg3 ; Вывод на экран строки
22 call sprintf ; 'Сообщение № 3'
23
24 _end:
25 call quit ; вызов подпрограммы завершения
```

Рис. 2.1: Файл lab7-1.asm

```
%include 'in_out.asm'

SECTION .data

msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0

SECTION .text

GLOBAL _start
```

```

_start:
jmp _label2

_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'

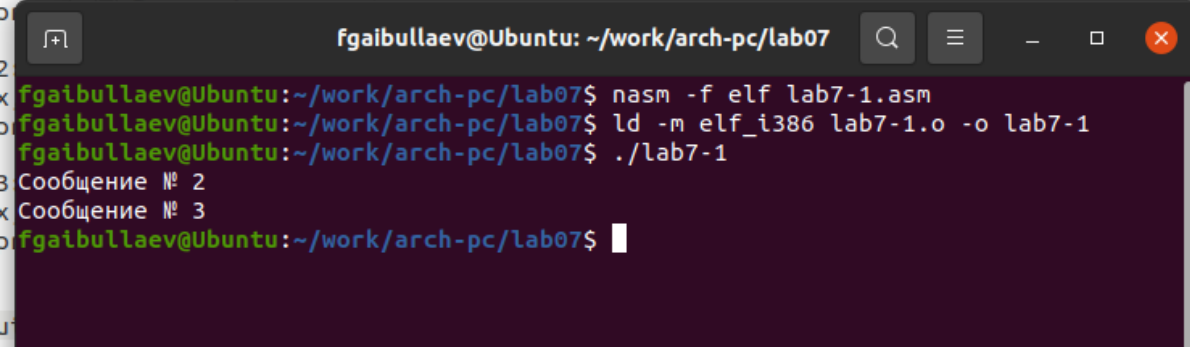
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'

_end:
call quit ; вызов подпрограммы завершения

```

Создайте исполняемый файл и запустите его.



The screenshot shows a terminal window with the title 'fgaibullaev@Ubuntu: ~/work/arch-pc/lab07'. The user has entered the following commands:

```

fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ ./lab7-1

```

The output of the program is visible in the terminal:

```

Сообщение № 2
Сообщение № 3

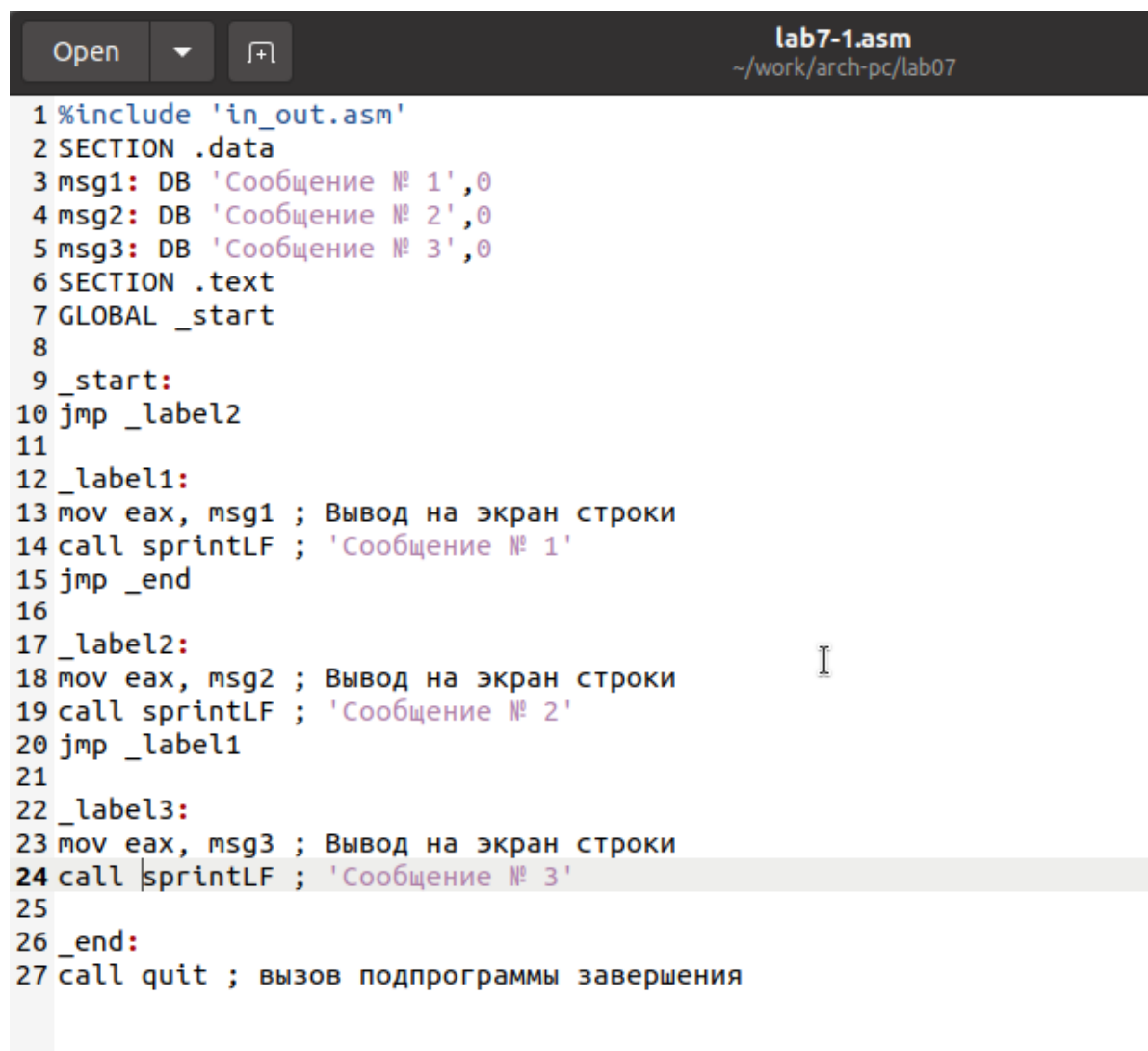
```

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`). Измените 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 ; Вывод на экран строки
14 call sprintfLF ; 'Сообщение № 1'
15 jmp _end
16
17 _label2:
18 mov eax, msg2 ; Вывод на экран строки
19 call sprintfLF ; 'Сообщение № 2'
20 jmp _label1
21
22 _label3:
23 mov eax, msg3 ; Вывод на экран строки
24 call sprintfLF ; 'Сообщение № 3'
25
26 _end:
27 call quit ; вызов подпрограммы завершения
```

Рис. 2.3: Файл lab7-1.asm:

```
%include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
```

```
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
jmp _label2

_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'

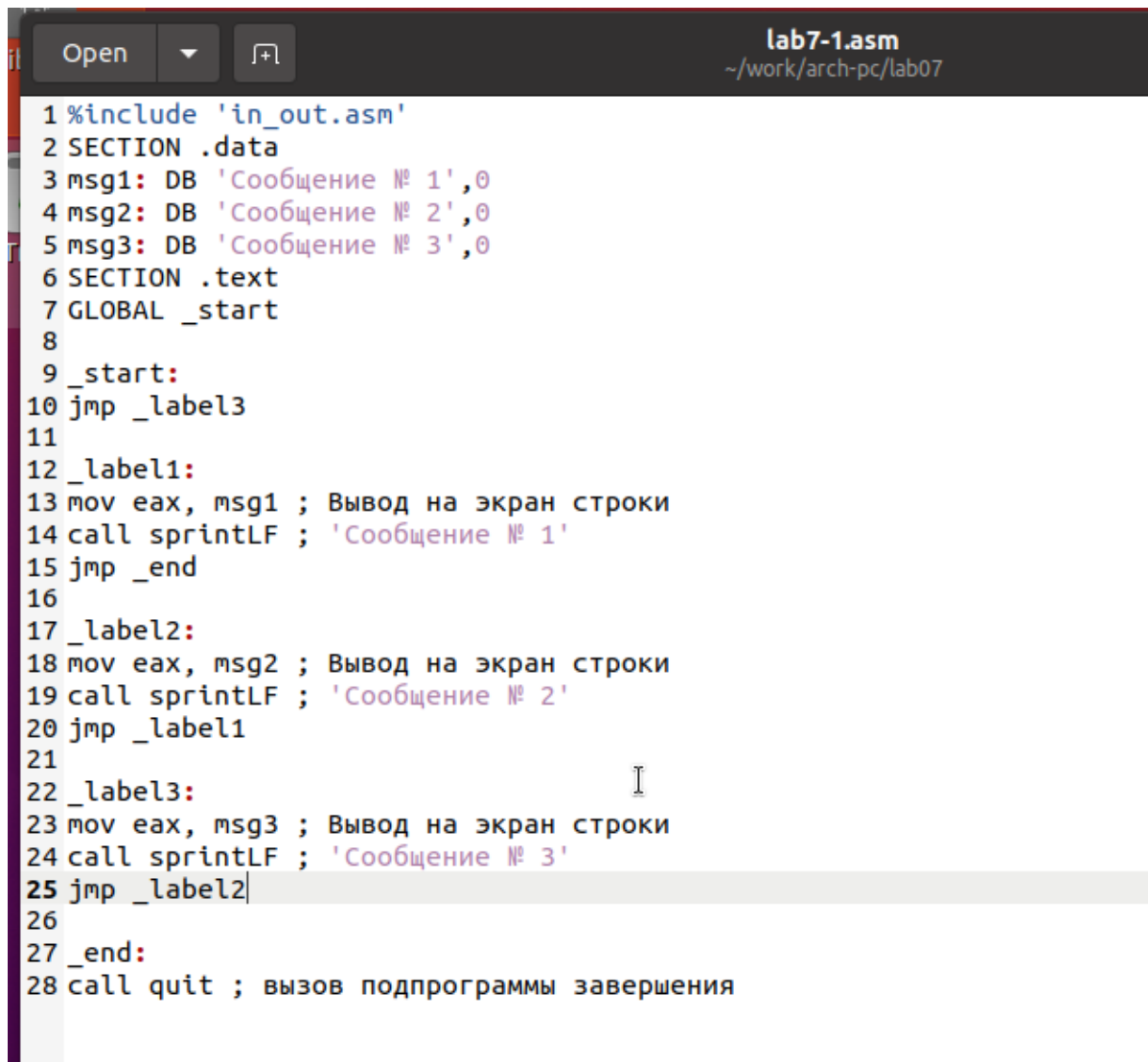
_end:
call quit ; вызов подпрограммы завершения
```

```
fgaibullaev@ubuntu:~/work/arch-pc/lab07$  
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm  
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_1386 lab7-1.o -o lab7-1  
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ ./lab7-1  
Сообщение № 2  
Сообщение № 1  
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.4: Запуск программы lab7-1.asm:

Измените текст программы добавив или 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 ; Вывод на экран строки
14 call sprintfLF ; 'Сообщение № 1'
15 jmp _end
16
17 _label2:
18 mov eax, msg2 ; Вывод на экран строки
19 call sprintfLF ; 'Сообщение № 2'
20 jmp _label1
21
22 _label3:
23 mov eax, msg3 ; Вывод на экран строки
24 call sprintfLF ; 'Сообщение № 3'
25 jmp _label2
26
27 _end:
28 call quit ; вызов подпрограммы завершения
```

Рис. 2.5: Файл lab7-1.asm

```
%include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
```

```

_start:
jmp _label3

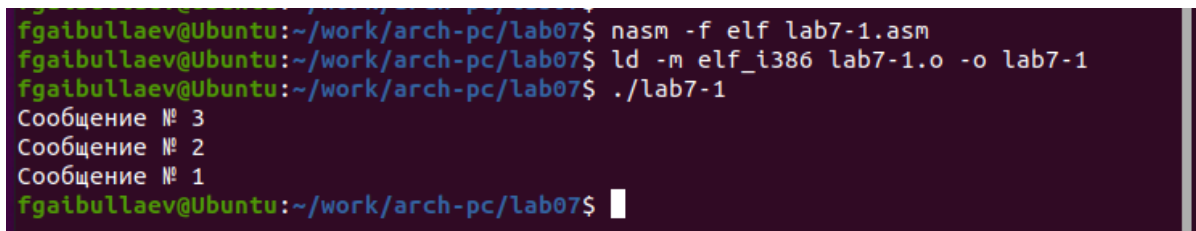
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2

_end:
call quit ; вызов подпрограммы завершения

```



```

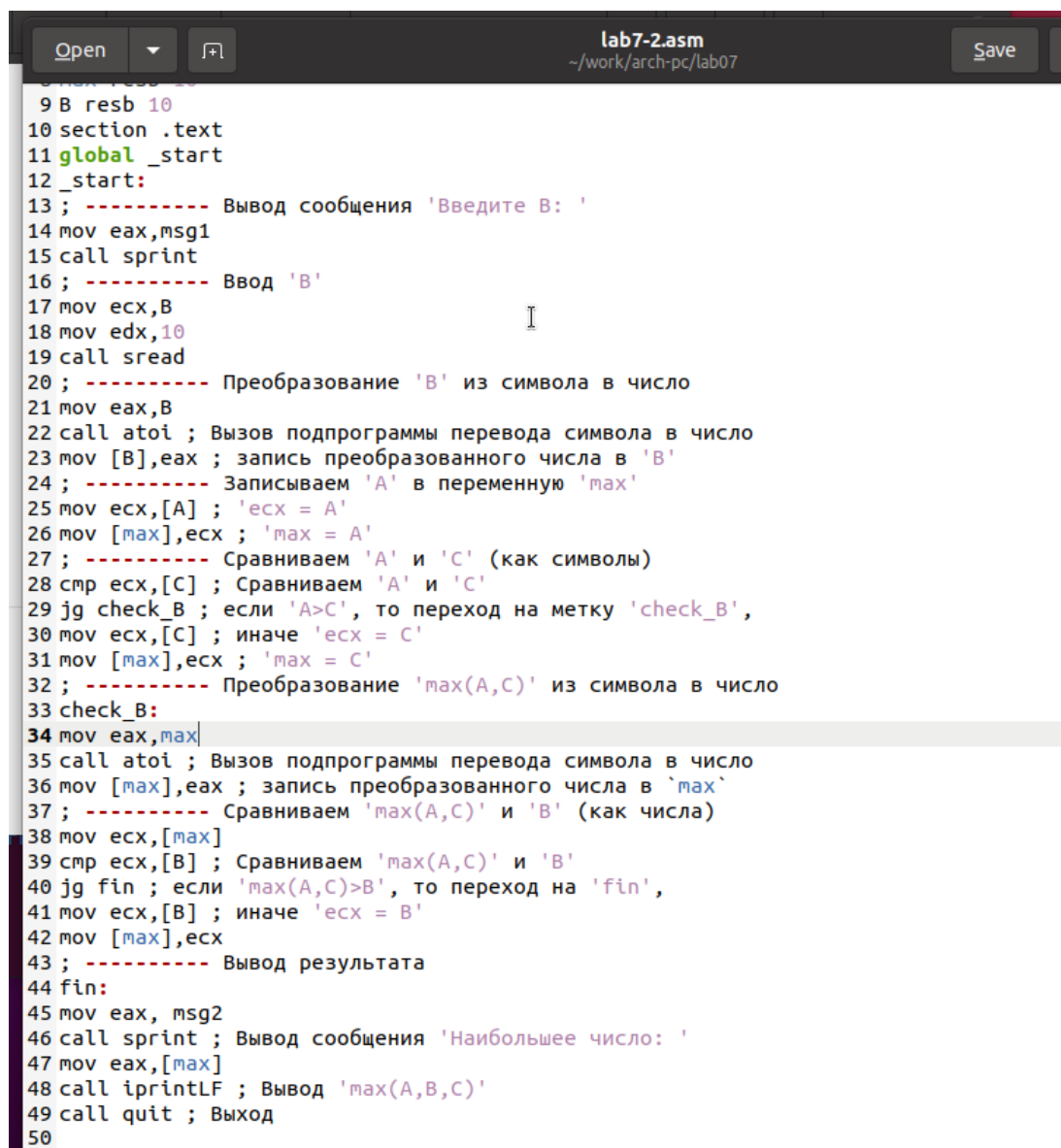
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$

```

Рис. 2.6: Запуск программы lab7-1.asm

3. Использование инструкции `jmp` приводит к переходу в любом случае. Од-

нако, 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: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры. Создайте исполняемый файл и проверьте его работу для разных значений В.



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

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 ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A] ; 'ecx = A'
26 mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 jg check_B ; если 'A>C', то переход на метку 'check_B',
30 mov ecx,[C] ; иначе 'ecx = C'
31 mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,[max]
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [max],eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 mov ecx,[max]
39 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 mov ecx,[B] ; иначе 'ecx = B'
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 mov eax, msg2
46 call sprint ; Вывод сообщения 'Наибольшее число: '
47 mov eax,[max]
48 call iprintLF ; Вывод 'max(A,B,C)'
49 call quit ; Выход
50
```

Рис. 2.7: Файл lab7-2.asm

```
%include 'in_out.asm'

section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
```

```

section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число

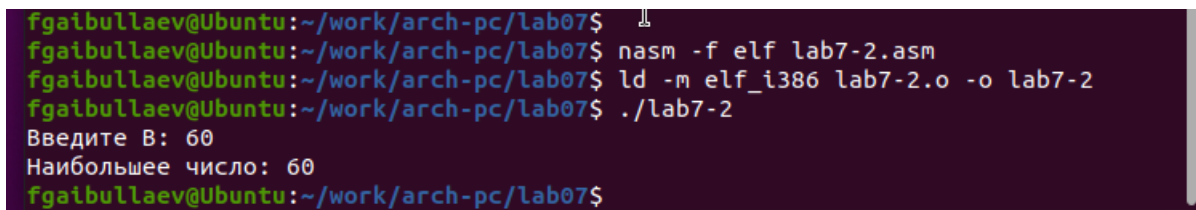
```



```

mov [max],eax ; запись преобразованного числа в `max`
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход

```



```

fgaibullaev@Ubuntu:~/work/arch-pc/lab07$
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
fgaibullaev@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. Создайте файл листинга для программы из файла lab7-2.asm

```

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

lab7-2.asm
lab7-2.lst

104 103 00000083 59 <1> pop ecx
105 104 00000084 58 <1> pop eax
106 105 00000085 C3 <1> ret
107 106 <1>
108 107 <1>
109 108 <1> ;----- iprintLF -----
110 109 <1> ; Функция вывода на экран чисел в формате ASCII
111 110 <1> ; входные данные: mov eax,<int>
112 111 <1> iprintLF:
113 112 00000086 E8C9FFFFFF <1> call iprint
114 113 <1>
115 114 0000008B 50 <1> push eax
116 115 0000008C B80A000000 <1> mov eax, 0Ah
117 116 00000091 50 <1> push eax
118 117 00000092 89E0 <1> mov eax, esp
119 118 00000094 E876FFFFFF <1> call sprintf
120 119 00000099 58 <1> pop eax
121 120 0000009A 58 <1> pop eax
122 121 0000009B C3 <1> ret
123 122 <1>
124 123 <1> ;----- atoi -----
125 124 <1> ; Функция преобразования ascii-код символа в целое
число
126 125 <1> ; входные данные: mov eax,<int>
127 126 <1> atoi:
128 127 0000009C 53 <1> push ebx
129 128 0000009D 51 <1> push ecx
130 129 0000009E 52 <1> push edx
131 130 0000009F 56 <1> push esi
132 131 000000A0 89C6 <1> mov esi, eax
133 132 000000A2 B800000000 <1> mov eax, 0
134 133 000000A7 B900000000 <1> mov ecx, 0
135 134 <1>
136 135 <1> .multiplyLoop:
137 136 000000AC 31DB <1> xor ebx, ebx
138 137 000000AE 8A1C0E <1> mov bl, [esi+ecx]
139 138 000000B1 80FB30 <1> cmp bl, 48
140 139 000000B4 7C14 <1> jl .finished
141 140 000000B6 80FB39 <1> cmp bl, 57
142 141 000000B9 7F0F <1> jg .finished

```

Рис. 2.9: Файл листинга lab7-2

Внимательно ознакомиться с его форматом и содержимым. Подробно объяснить содержимое трёх строк файла листинга по выбору.

строка 128

- 128 - номер строки
- 0000009C - адрес
- 53 - машинный код

- push ebx - код программы

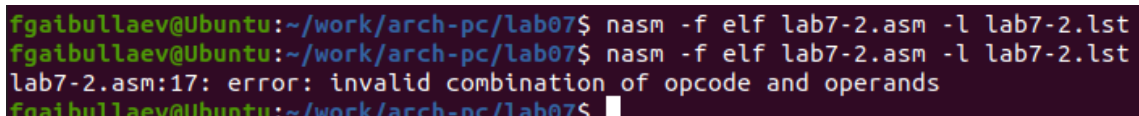
строка 129

- 129 - номер строки
- 0000009D - адрес
- 51 - машинный код
- push ecx - код программы

строка 130

- 130 - номер строки
- 0000009E - адрес
- 52 - машинный код
- push edx - код программы

Откройте файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалите один операнд. Выполните трансляцию с получением файла листинга



```
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:17: error: invalid combination of opcode and operands
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.10: ошибка трансляции lab7-2

```

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

lab7-2.asm
lab7-2.lst

175 3 00000012 00
176 4 00000013 D09DD0B0D0B8D0B1D0- msg2 db "Наибольшее число: ",0h
177 4 0000001C BED0BBD18CD188D0B5-
178 4 00000025 D0B520D187D0B8D181-
179 4 0000002E D0BBD0BE3A2000
180 5 00000035 32300000 A dd '20'
181 6 00000039 35300000 C dd '50'
182 7
183 8 00000000 <res 0000000A> section .bss
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 ; ----- Ввод 'B'
193 17 mov ecx,
194 18 000000F2 BA0A000000 error: invalid combination of opcode and operands
195 19 000000F7 E847FFFFFF mov edx,10
196 20 call sread
197 21 000000FC B8[0A000000] ; ----- Преобразование 'B' из символа в число
198 22 00000101 E896FFFFFF mov eax,B
199 23 00000106 A3[0A000000] call atoi ; Вызов подпрограммы перевода символа в
200 24 число
201 25 0000010B 8B0D[35000000] mov [B],eax ; запись преобразованного числа в 'B'
202 26 00000111 890D[00000000] ; ----- Записываем 'A' в переменную 'max'
203 27 mov ecx,[A] ; 'ecx = A'
204 28 00000117 3B0D[39000000] mov [max],ecx ; 'max = A'
205 29 0000011D 7F0C ; ----- Сравниваем 'A' и 'C' (как символы)
206 30 0000011F 8B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
207 31 00000125 890D[00000000] jg check_B ; если 'A>C', то переход на метку
208 32 'check_B',
209 33 mov ecx,[C] ; иначе 'ecx = C'
210 34 0000012B B8[00000000] mov [max],ecx ; 'max = C'
211 35 00000130 E867FFFFFF ; ----- Преобразование 'max(A,C)' из символа в
212 36 число
213 37 check_B:
214 38 mov eax,max
215 39 call atoi ; Вызов подпрограммы перевода символа в
216 40 число

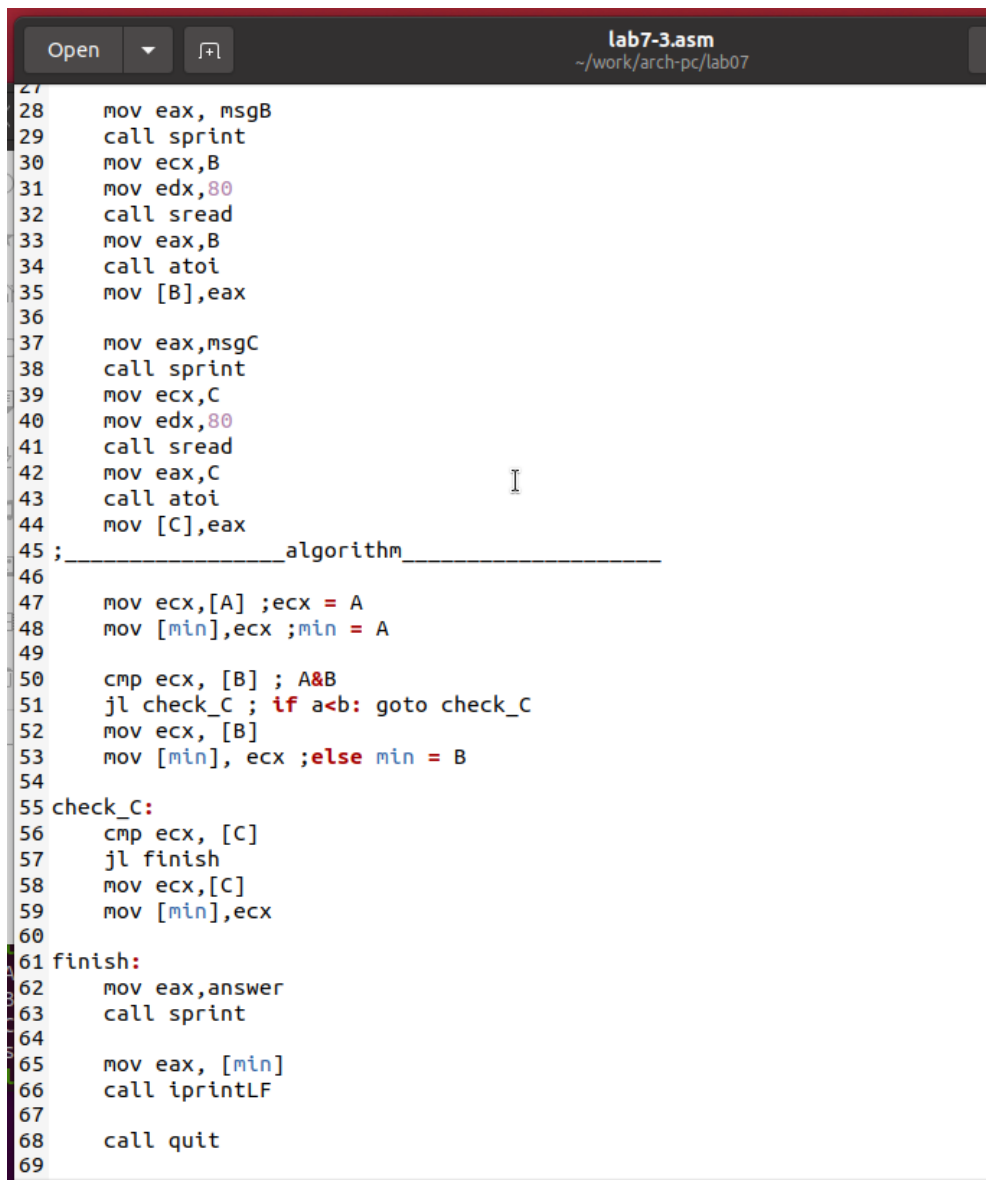
```

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 18 - 83, 73, 30



```
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 ;-----algorithm-----
46
47     mov ecx, [A] ;ecx = A
48     mov [min], ecx ;min = A
49
50     cmp ecx, [B] ; A&B
51     jl check_C ; if a<b: goto check_C
52     mov ecx, [B]
53     mov [min], ecx ;else min = B
54
55 check_C:
56     cmp ecx, [C]
57     jl finish
58     mov ecx, [C]
59     mov [min], ecx
60
61 finish:
62     mov eax, answer
63     call sprint
64
65     mov eax, [min]
66     call iprintLF
67
68     call quit
69
```

Рис. 2.12: Файл lab7-3.asm

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msgA:    DB 'Input A: ',0
```

```
msgB:          DB 'Input B: ',0
```

```
msgC:    DB 'Input C: ',0
```

```
answer: DB 'Smallest: ',0
```

```

SECTION .bss
    A:  RESB 80
    B:  RESB 80
    C:  RESB 80
    result:      RESB 80
    min: RESB 80

```

```

SECTION .text
    GLOBAL _start

```

```

_start:
    mov eax,msgA
    call sprint
    mov ecx,A
    mov edx,80
    call sread
    mov eax,A
    call atoi
    mov [A],eax

    mov eax, msgB
    call sprint
    mov ecx,B
    mov edx,80
    call sread
    mov eax,B
    call atoi
    mov [B],eax

```

```

    mov eax,msgC
    call sprint
    mov ecx,C
    mov edx,80
    call sread
    mov eax,C
    call atoi
    mov [C],eax
;_____algorithm_____

    mov ecx,[A] ;ecx = A
    mov [min],ecx ;min = A

    cmp ecx, [B] ; A&B
    jl check_C ; if a<b: goto check_C
    mov ecx, [B]
    mov [min], ecx ;else min = B

check_C:
    cmp ecx, [C]
    jl finish
    mov ecx,[C]
    mov [min],ecx

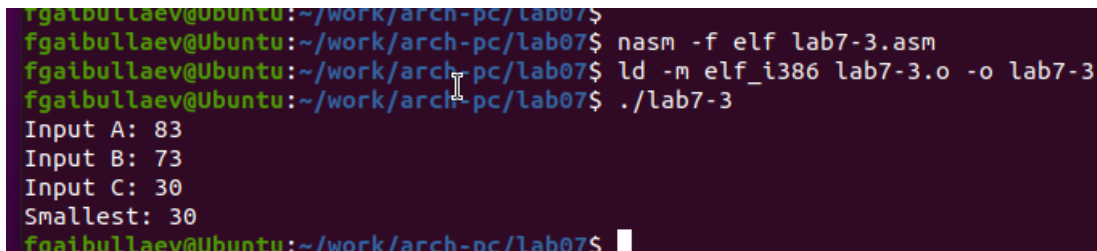
finish:
    mov eax,answer
    call sprint

```

```
mov eax, [min]
```

```
call iprintLF
```

```
call quit
```



```
fgaibullaev@ubuntu:~/work/arch-pc/lab07$  
fgaibullaev@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm  
fgaibullaev@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3  
fgaibullaev@ubuntu:~/work/arch-pc/lab07$ ./lab7-3  
Input A: 83  
Input B: 73  
Input C: 30  
Smallest: 30  
fgaibullaev@ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.13: Запуск программы lab7-3.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 18

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



```

27     mov edx,80
28     call sread
29     mov eax,X
30     call atoi
31     mov [X],eax
32 ; _____algorithm_____
33
34     mov edx, [A]
35     mov ebx,1
36     cmp ebx, edx
37     jne first
38     jmp second
39
40 first:
41     mov eax,[A]
42     mov ebx,[A]
43     mul ebx
44     call iprintLF
45     call quit
46 second:
47     mov eax,[X]
48     add eax,10
49     call iprintLF
50     call quit
51

```

Рис. 2.14: Файл lab7-4.asm

```

#include 'in_out.asm'

SECTION .data
    msgA:  DB 'Input A: ',0
    msgX:   DB 'Input X: ',0

SECTION .bss
    A:  RESB 80
    X:  RESB 80
    result:  RESB 80

SECTION .text

```

```

GLOBAL _start

_start:
    mov eax,msgA
    call sprint
    mov ecx,A
    mov edx,80
    call sread
    mov eax,A
    call atoi
    mov [A],eax

    mov eax,msgX
    call sprint
    mov ecx,X
    mov edx,80
    call sread
    mov eax,X
    call atoi
    mov [X],eax

;-----algorithm-----

    mov edx, [A]
    mov ebx,1
    cmp ebx, edx
    jne first
    jmp second

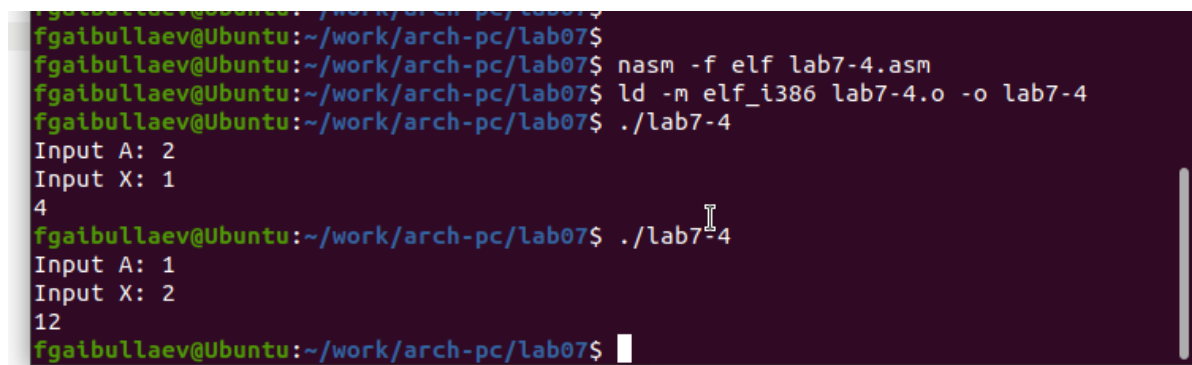
first:

```

```
mov eax,[A]
mov ebx,[A]
mul ebx
call iprintLF
call quit
```

second:

```
mov eax,[X]
add eax,10
call iprintLF
call quit
```

A terminal window with a dark purple background. The prompt is 'fgaibullaev@Ubuntu:~/work/arch-pc/lab07\$'. The user enters 'nasm -f elf lab7-4.asm', then 'ld -m elf\_i386 lab7-4.o -o lab7-4', and finally './lab7-4'. The program outputs 'Input A: 2', 'Input X: 1', and '4'. The user then runs './lab7-4' again, and the program outputs 'Input A: 1', 'Input X: 2', and '12'.

```
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ ./lab7-4
Input A: 2
Input X: 1
4
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$ ./lab7-4
Input A: 1
Input X: 2
12
fgaibullaev@Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.15: Запуск программы lab7-4.asm

## 3 Выводы

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