

Отчёт по лабораторной работе 7

Архитектура компьютера

Арслан Юсупов

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Самостоятельное задание	15
3	Выводы	20

Список иллюстраций

2.1	Создал каталог и файл	6
2.2	Программа в файле lab7-1.asm	7
2.3	Запуск программы lab7-1.asm	8
2.4	Программа в файле lab7-1.asm	8
2.5	Запуск программы lab7-1.asm	9
2.6	Программа в файле lab7-1.asm	10
2.7	Запуск программы lab7-1.asm	10
2.8	Программа в файле lab7-2.asm	12
2.9	Запуск программы lab7-2.asm	12
2.10	Файл листинга lab7-2	13
2.11	Ошибка трансляции lab7-2	14
2.12	Файл листинга с ошибкой lab7-2	15
2.13	Программа в файле prog1.asm	16
2.14	Запуск программы prog1.asm	17
2.15	Программа в файле prog2.asm	18
2.16	Запуск программы prog2.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

Создал каталог для программ лабораторной работы № 7 и файл lab7-1.asm.
(рис. 2.1)

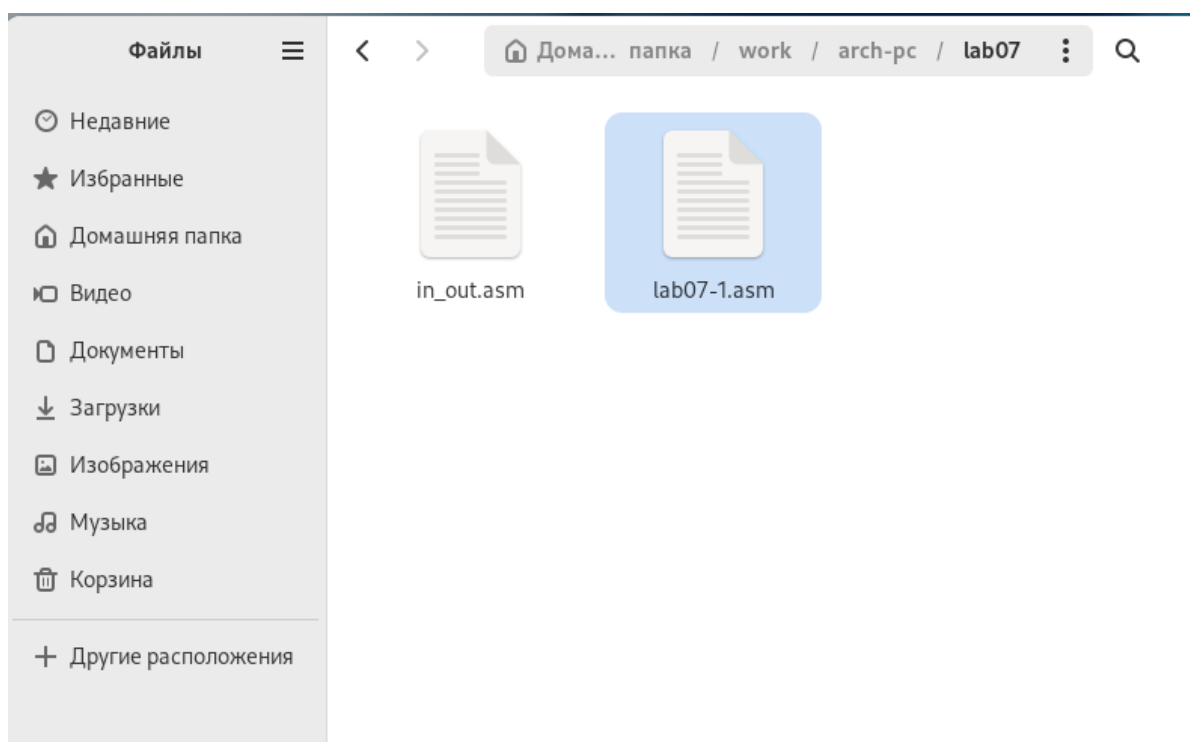



Рис. 2.1: Создал каталог и файл

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. (рис. 2.2)



```
Открыть ▾ + lab7-1.asm
~/.work/arch-pc/lab07

%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
call sprintLF

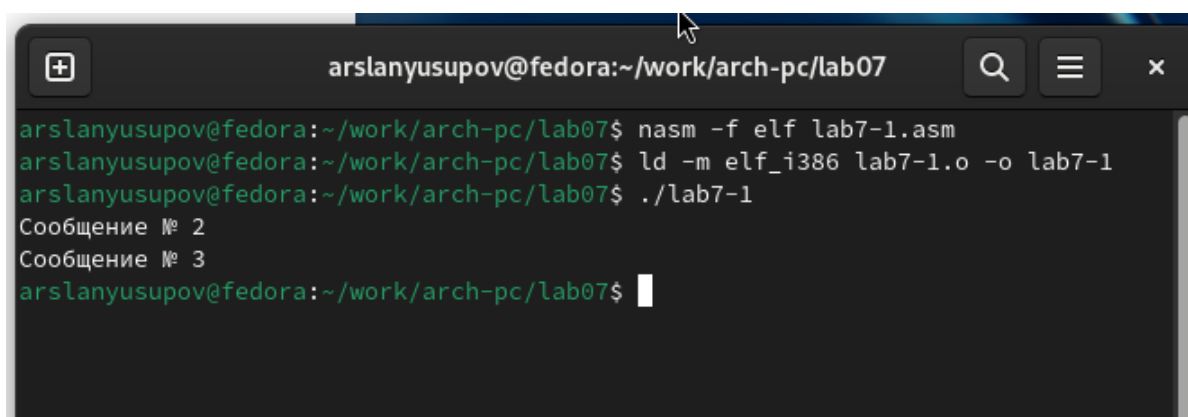
_label2:
mov eax, msg2
call sprintLF

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.2: Программа в файле lab7-1.asm

Создал исполняемый файл и запустил его. (рис. 2.3)

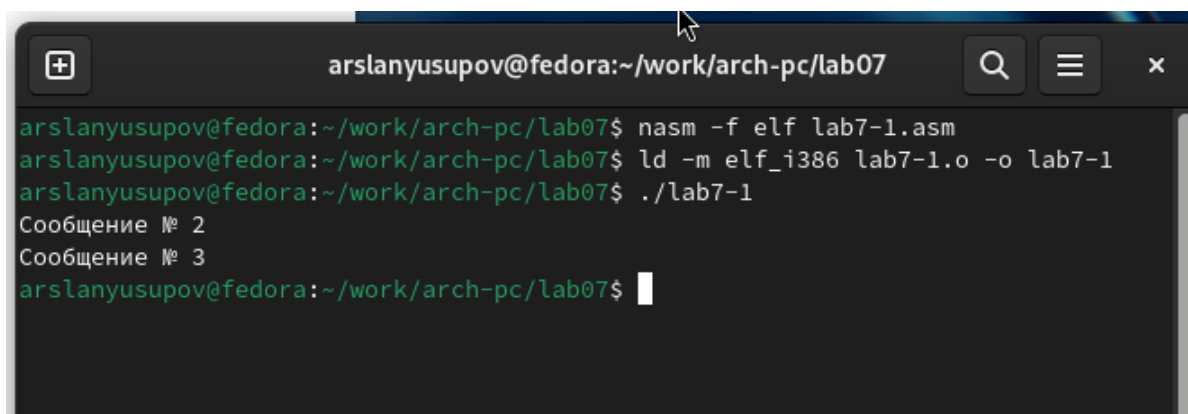
A terminal window titled 'arslanyusupov@fedora:~/work/arch-pc/lab07' with search, menu, and close buttons. The terminal shows the following commands and output:

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

Рис. 2.3: 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. (рис. 2.4) (рис. 2.5)

A terminal window titled 'arslanyusupov@fedora:~/work/arch-pc/lab07' with search, menu, and close buttons. The terminal shows the following commands and output:

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

Рис. 2.4: Программа в файле lab7-1.asm



```
Открыть ▾ + lab7-1.asm
~/work/arch-pc/lab07

%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
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.5: Запуск программы lab7-1.asm

Изменил текст программы, изменив инструкции jmp, чтобы вывод программы был следующим (рис. 2.6) (рис. 2.7):

Сообщение № 3
Сообщение № 2
Сообщение № 1



```
Открыть ▾ + lab7-1.asm
~/work/arch-pc/lab07

%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

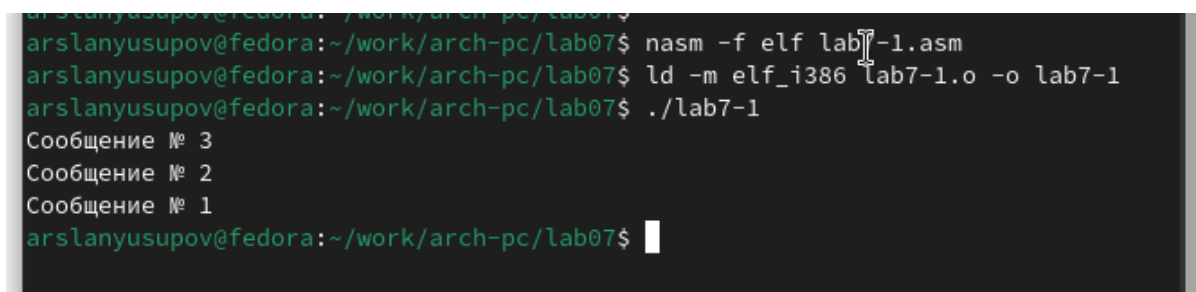
_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit
```

Рис. 2.6: Программа в файле lab7-1.asm

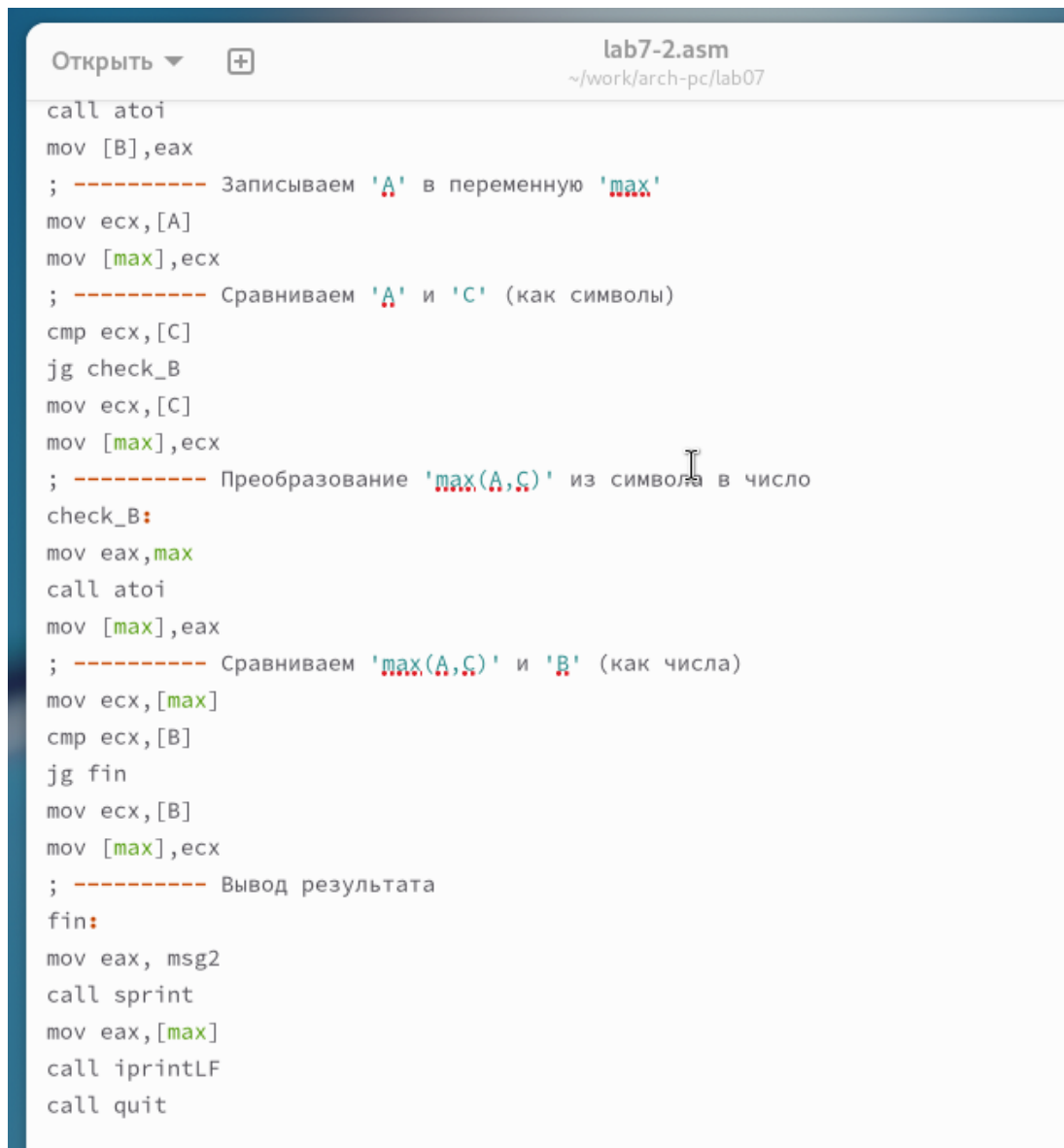


```
arslanyusupov@fedora: ~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
arslanyusupov@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
arslanyusupov@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
arslanyusupov@fedora:~/work/arch-pc/lab07$
```

Рис. 2.7: 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: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры.

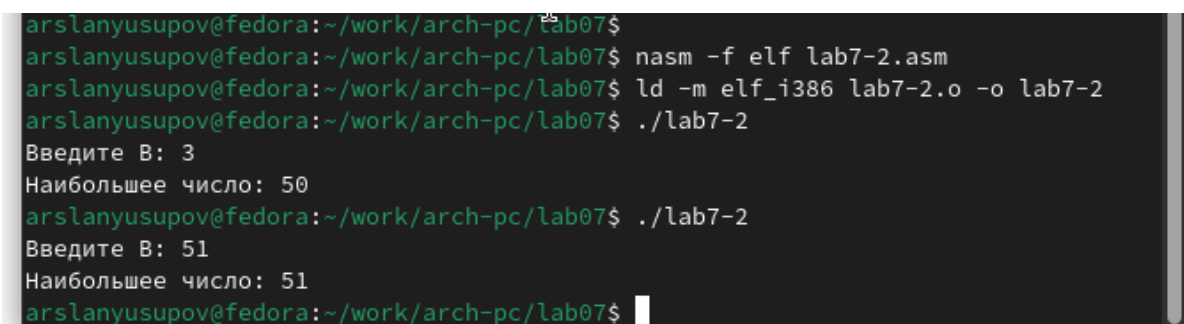
Создал исполняемый файл и проверил его работу для разных значений В (рис. 2.8) (рис. 2.9).



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

call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A]
mov [max],ecx
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi
mov [max],eax
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint
mov eax,[max]
call iprintLF
call quit
```

Рис. 2.8: Программа в файле lab7-2.asm



```
arslanyusupov@fedora:~/work/arch-pc/lab07$
arslanyusupov@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
arslanyusupov@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
arslanyusupov@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 3
Наибольшее число: 50
arslanyusupov@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 51
Наибольшее число: 51
arslanyusupov@fedora:~/work/arch-pc/lab07$
```

Рис. 2.9: Запуск программы lab7-2.asm

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` (рис. 2.10)

```

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. Ah>      max resb 10
184      9 0000000A <res. Ah>      B resb 10
185     10
186     11      section .text
187     12      global _start
188     13      _start:
189     14      ; ----- Вывод сообщения 'Введите B: '
190     15      mov eax, msg1
191     16      call sprintf
192     17      ; ----- Ввод 'B'
193     18      mov ecx, B
194     19      mov edx, 10
195     20      call sread
196     21      ; ----- Преобразование 'B' из символа в
197     22      число
198     23      mov eax, B
199     24      call atoi
200     25      mov [B], eax
201     26      ; ----- Записываем 'A' в переменную 'max'
202     27      mov ecx, [A]
203     28      mov [max], ecx
204     29      ; ----- Сравниваем 'A' и 'C' (как символы)
205     30      cmp ecx, [C]
206     31      jg check_B
207     32      mov ecx, [C]
208     33      mov [max], ecx

```

Рис. 2.10: Файл листинга `lab7-2`

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

строка 189

- 14 - номер строки в подпрограмме
- 000000E8 - адрес
- B8[00000000] - машинный код
- mov eax,msg1 - код программы - перекладывает msg1 в eax

строка 190

- 15 - номер строки в подпрограмме
- 000000ED - адрес
- E81DFFFFFF - машинный код
- call sprint - код программы - вызов подпрограммы печати

строка 192

- 17 - номер строки в подпрограмме
- 000000F2 - адрес
- B9[0A000000] - машинный код
- mov ecx,B - код программы - перекладывает B в ecx

Открыл файл с программой lab7-2.asm и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга. (рис. 2.11) (рис. 2.12)

```

arslanyusupov@fedora:~/work/arch-pc/lab07$
arslanyusupov@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
arslanyusupov@fedora:~/work/arch-pc/lab07$
arslanyusupov@fedora:~/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
arslanyusupov@fedora:~/work/arch-pc/lab07$

```

Рис. 2.11: Ошибка трансляции lab7-2

```

lab7-2.asm
lab7-2.lst

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 F862FFFFFF 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,
215 39 *****
      error: invalid combination of opcode and
      operands
216 40 00000145 7F0C ig 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 F8B2FFFFFF call sprint
223 47 0000015D A1[00000000] mov eax,[max]
224 48 00000162 E81FFFFFFF call iprintLF
225 49 00000167 E86FFFFFFF call quit

```

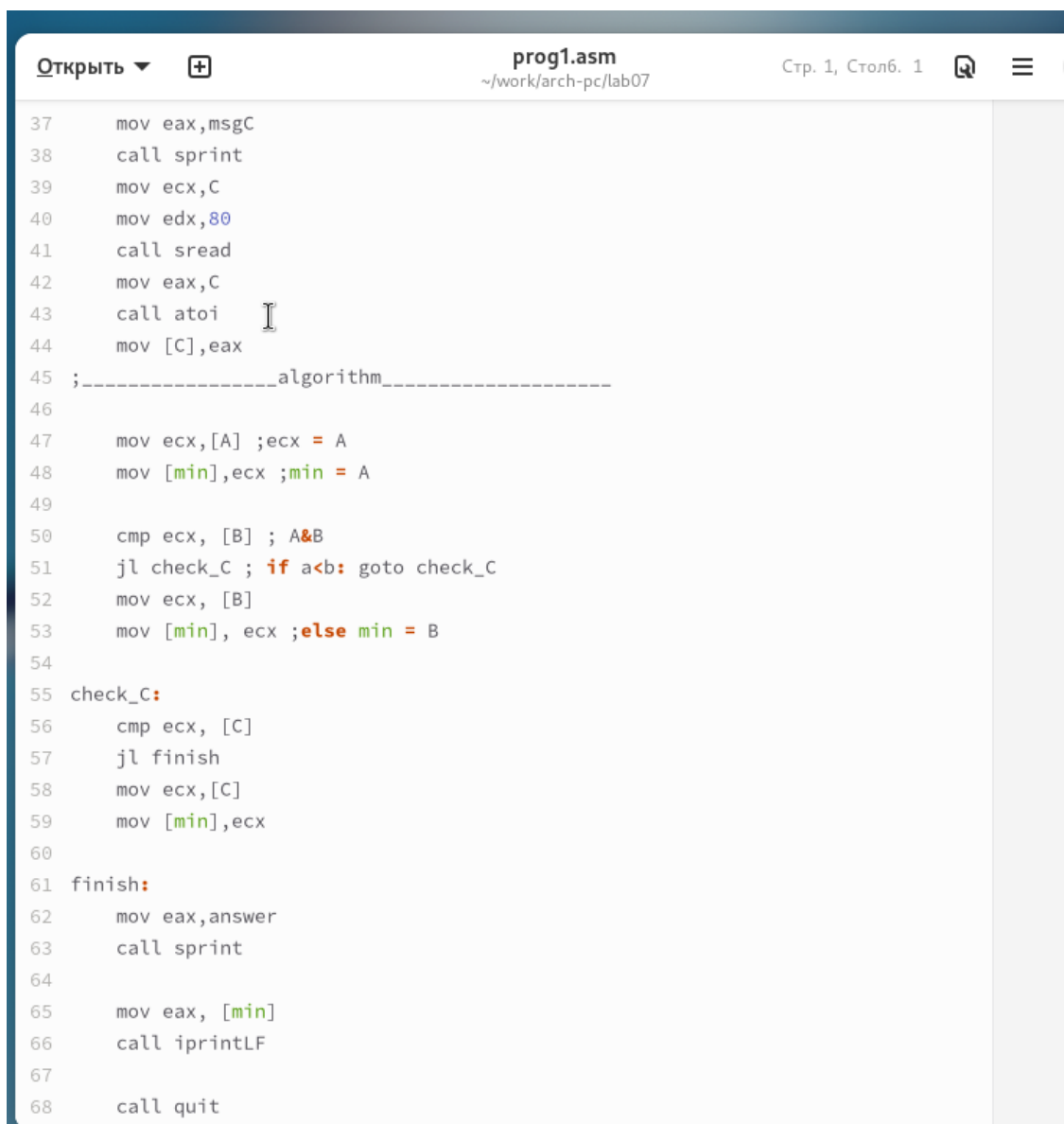
Рис. 2.12: Файл листинга с ошибкой lab7-2

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

2.1 Самостоятельное задание

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 (рис. 2.13) (рис. 2.14) для варианта 2 - 82,59,61



```
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
```

Рис. 2.13: Программа в файле prog1.asm


```
arslanyusupov@fedora:~/work/arch-pc/lab07$  
arslanyusupov@fedora:~/work/arch-pc/lab07$ nasm -f elf prog1.asm  
arslanyusupov@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 prog1.o -o prog1  
arslanyusupov@fedora:~/work/arch-pc/lab07$ ./prog1  
Input A: 82  
Input B: 59  
Input C: 61  
Smallest: 59  
arslanyusupov@fedora:~/work/arch-pc/lab07$
```

Рис. 2.14: Запуск программы prog1.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. (рис. 2.15) (рис. 2.16)

для варианта 2

$$\begin{cases} a - 1, x < a \\ x - 1, x \geq a \end{cases}$$

Если подставить $x = 5, a = 7$ получается $7 - 1 = 6$.

Если подставить $x = 6, a = 4$ получается $6 - 1 = 5$.

Открыть ▾ 

prog2.asm
~/work/arch-pc/lab07

Стр. 47, Столб. 14

```
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 ;-----algorithm-----
33
34     mov ebx, [X]
35     mov edx, [A]
36     cmp ebx, edx
37     jb first
38     jmp second
39
40 first:
41     mov eax,[A]
42     sub eax,1
43     call iprintLF
44     call quit
45 second:
46     mov eax,[X]
47     sub eax,1
48     call iprintLF
49     call quit
50
51
```

Рис. 2.15: Программа в файле prog2.asm

```
arslanyusupov@fedora:~/work/arch-pc/lab07$  
arslanyusupov@fedora:~/work/arch-pc/lab07$ nasm -f elf prog2.asm  
arslanyusupov@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 prog2.o -o prog2  
arslanyusupov@fedora:~/work/arch-pc/lab07$ ./prog2  
Input A: 7  
Input X: 5  
6  
arslanyusupov@fedora:~/work/arch-pc/lab07$ ./prog2  
Input A: 4  
Input X: 6  
5  
arslanyusupov@fedora:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы prog2.asm

3 Выводы

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