

# **Отчёт по лабораторной работе 7**

**Архитектура компьютера**

Бегенджов Гурбанмырат

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Самостоятельное задание . . . . .	16
<b>3</b>	<b>Выводы</b>	<b>21</b>

## 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 . . . . .	7
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 . . . . .	13
2.10	Файл листинга lab7-2 . . . . .	14
2.11	Ошибка трансляции lab7-2 . . . . .	15
2.12	Файл листинга с ошибкой lab7-2 . . . . .	16
2.13	Программа в файле prog1.asm . . . . .	17
2.14	Запуск программы prog1.asm . . . . .	17
2.15	Программа в файле prog2.asm . . . . .	19
2.16	Запуск программы prog2.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

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

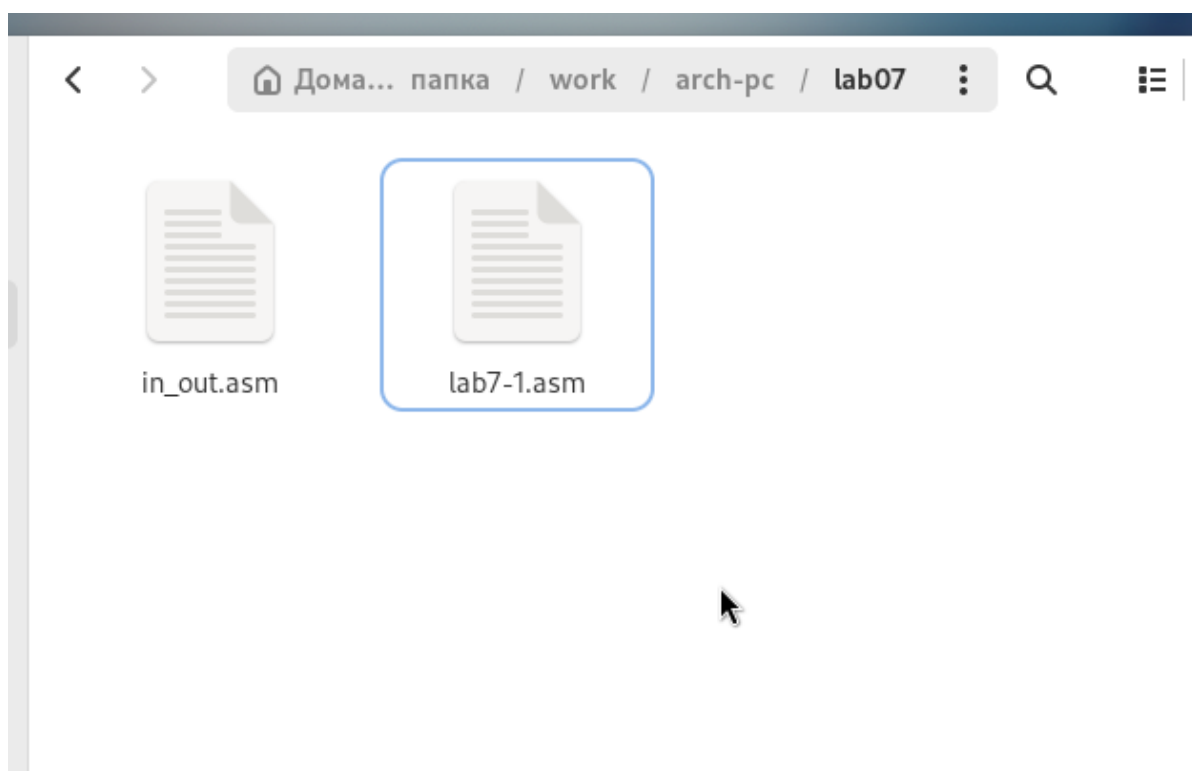


Рис. 2.1: Создал каталог и файл

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

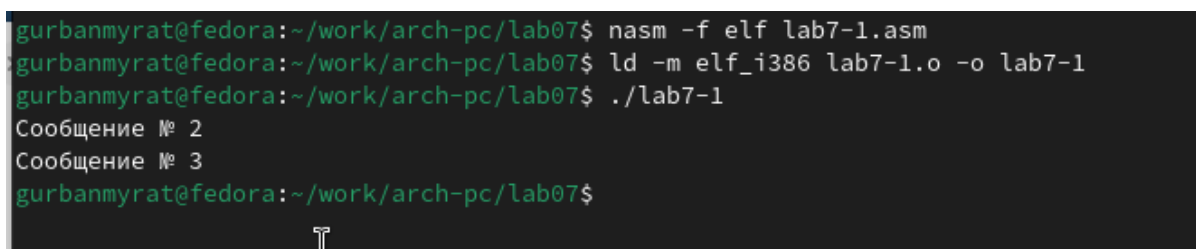
_label2:
mov eax, msg2
call sprintLF

_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)

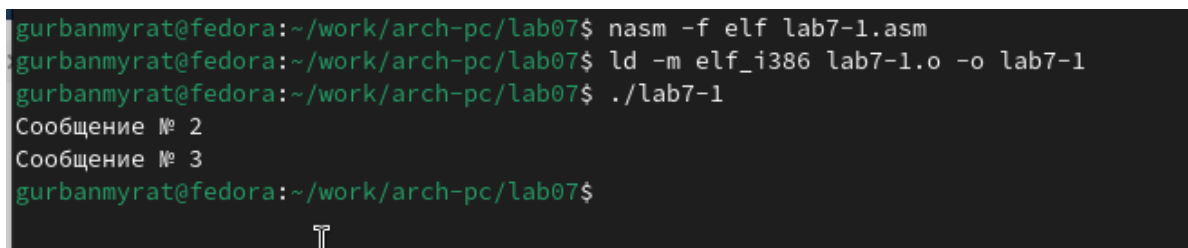


```
gurbanmyrat@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
gurbanmyrat@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
gurbanmyrat@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
gurbanmyrat@fedora:~/work/arch-pc/lab07$
```

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)



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

Рис. 2.4: Программа в файле 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
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF

_end:
call quit

```

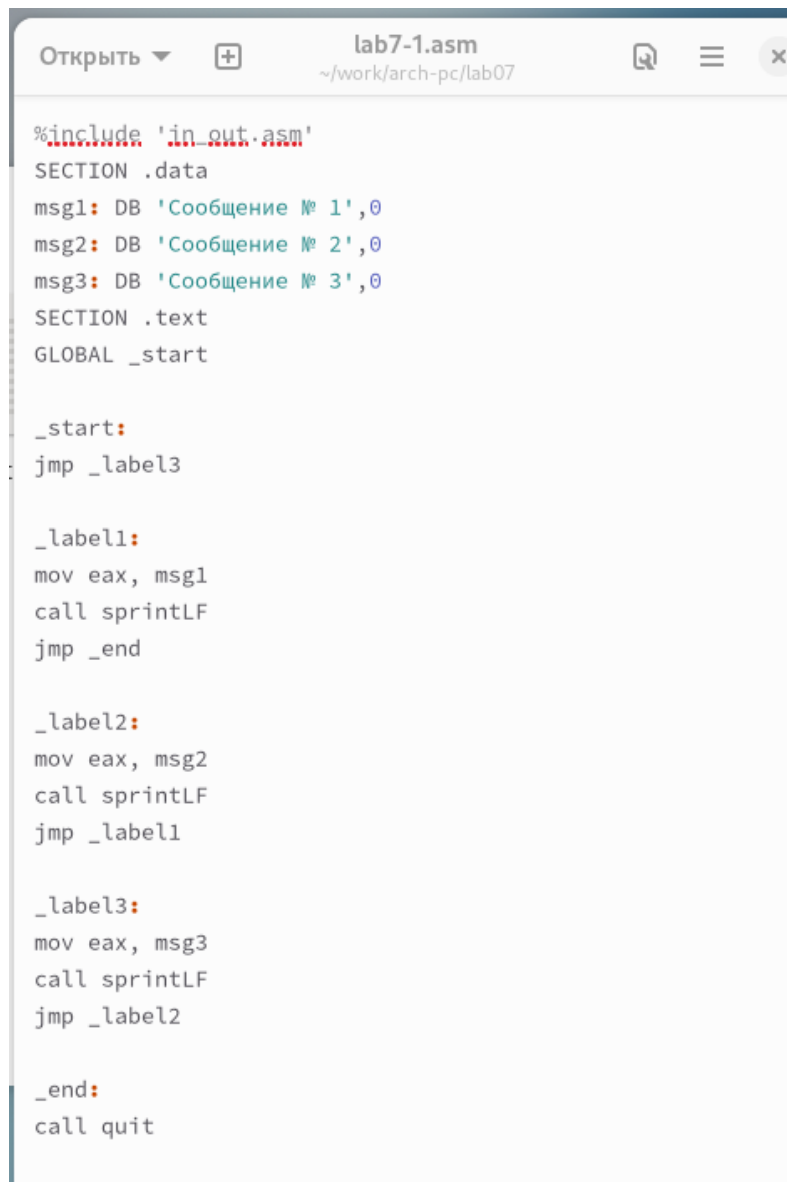
Рис. 2.5: Запуск программы lab7-1.asm

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

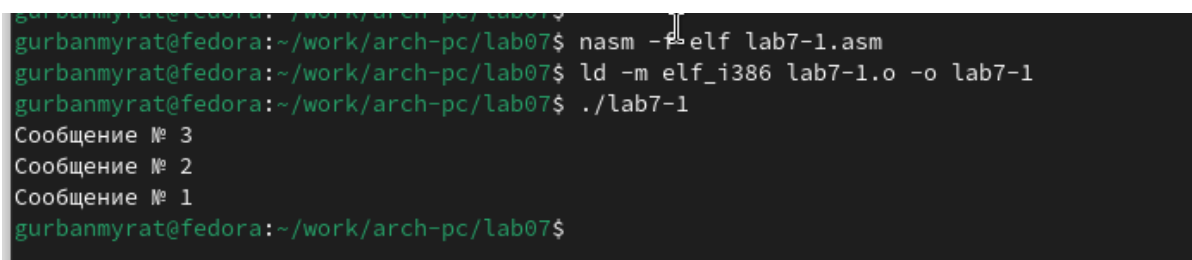
_label1:
mov eax, msg1
call sprintLF
jmp _end

_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

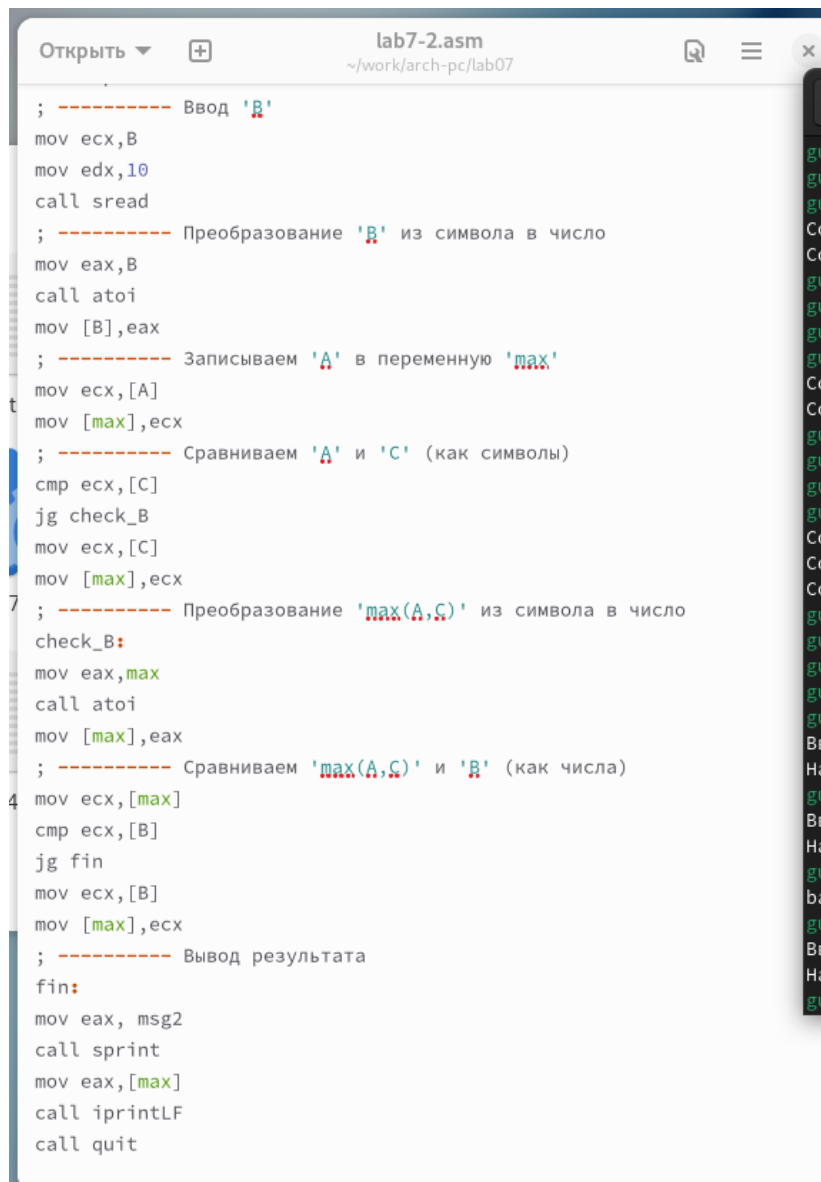


```
gurbanmyrat@fedora: ~/work/arch-pc/lab07$
gurbanmyrat@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
gurbanmyrat@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
gurbanmyrat@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
gurbanmyrat@fedora:~/work/arch-pc/lab07$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

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).



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

; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
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

```
gurbanmyrat@fedora:~/work/arch-pc/lab07$  
gurbanmyrat@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm  
gurbanmyrat@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2  
gurbanmyrat@fedora:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 10  
Наибольшее число: 50  
gurbanmyrat@fedora:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 40  
Наибольшее число: 50  
gurbanmyrat@fedora:~/work/arch-pc/lab07$ ./lab7-260  
bash: ./lab7-260: Нет такого файла или каталога  
gurbanmyrat@fedora:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 60  
Наибольшее число: 60  
gurbanmyrat@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)

```

170 000000E5 CDB0          <1>    int    80h
171 000000E7 C3          <1>    ret
      2
      section .data
3 00000000 D02D0B2D0B5D0B4D0- msg1 db 'Введите B: ',0h
3 00000009 B8D182D0B520423A20-
3 00000012 00
4 00000013 D02D0B2D0B5D0B4D0- msg2 db "Наибольшее число: ",0h
4 0000001C BFD0BBD18CD18BD0B5-
4 00000025 D0B520D187D0B8D181-
4 0000002F D0BBD0B52A7000
5 00000035 32300000      A dd '20'
6 00000039 35300000      C dd '50'
      7
      section .bss
8 00000000 <res Ah>      max resb 10
9 0000000A <res Ah>      B resb 10
10
      section .text
11      global _start
12      _start:
13      ; ----- Вывод сообщения 'Введите B: '
14 000000E8 B8[00000000]      mov eax,msg1
15 000000ED EB1DFFFFFF      call sprintf
16      ; ----- Ввод 'B'
17 000000F2 B9[0A000000]      mov ecx,B
18 000000F7 BA0A000000      mov edx,10
19 000000FC E842FFFFFF      call sread
20      ; ----- Преобразование 'B' из символа в число
21 00000101 B8[0A000000]      mov eax,B
22 00000106 EB01FFFFFF      call atoi
23 0000010B A3[0A000000]      mov [B],eax
24      ; ----- Записываем 'A' в переменную 'max'
25 00000110 BBD0[35000000]      mov ecx,[A]
26 00000116 B90D[00000000]      mov [max],ecx
27      ; ----- Сравниваем 'A' и 'C' (как символы)

```

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)

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

Рис. 2.11: Ошибка трансляции lab7-2

```

27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000] cmp ecx,[C]
29 00000122 7F0C ig check_B
30 00000124 8B0D[39000000] mov ecx,[C]
31 0000012A 8B0D[00000000] mov [max],ecx
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 00000130 B8[00000000] mov eax,max
35 00000135 E862FFFFFF call atoi
36 0000013A A3[00000000] mov [max],eax
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013F 8B0D[00000000] mov ecx,[max]
39 00000140 8B0D[00000000] cmp ecx,[B]
39 ***** error: invalid combination of opcode and operands
40 00000145 7F0C ig fin
41 00000147 8B0D[0A000000] mov ecx,[B]
42 0000014D 8B0D[00000000] mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 00000153 B8[13000000] mov eax,msg2
46 00000158 E8B2FFFFFF call sprintf
47 0000015D A1[00000000] mov eax,[max]
48 00000162 E81FFFFFFF call inprintf
49 00000167 E86FFFFFFF call quit

```

Рис. 2.12: Файл листинга с ошибкой lab7-2

Объектный файл не смог создаваться 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



```

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

mov ecx,[A]
mov [min],ecx

cmp ecx, [B]
jl check_C
mov ecx, [B]
mov [min], ecx

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

```

Рис. 2.13: Программа в файле prog1.asm

```

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

```

Рис. 2.14: Запуск программы prog1.asm

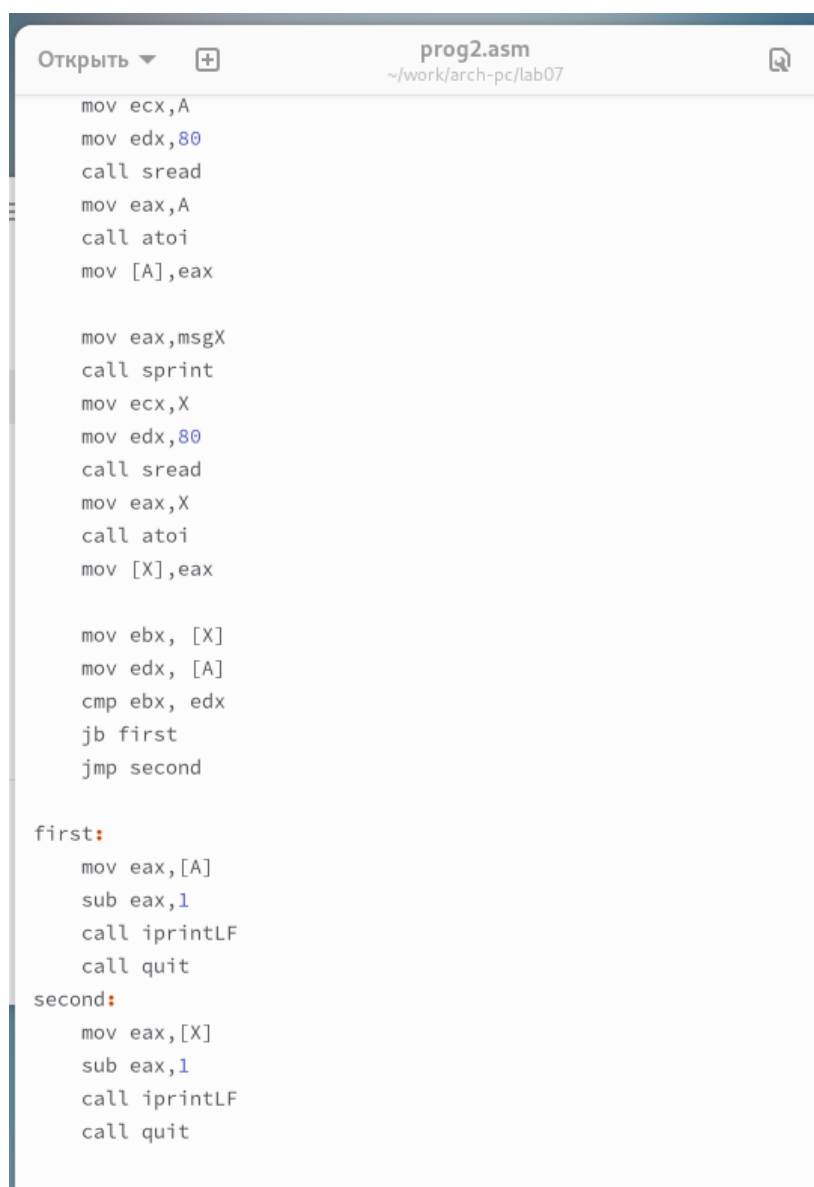
Напишите программу, которая для введенных с клавиатуры значений  $x$  и  $a$  вычисляет значение заданной функции  $f(x)$  и выводит результат вычислений. Вид функции  $f(x)$  выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте 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

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

mov ebx, [X]
mov edx, [A]
cmp ebx, edx
jb first
jmp second

first:
mov eax,[A]
sub eax,1
call iprintLF
call quit

second:
mov eax,[X]
sub eax,1
call iprintLF
call quit
```

Рис. 2.15: Программа в файле prog2.asm

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

Рис. 2.16: Запуск программы prog2.asm

## 3 Выводы

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