

# **Отчёт по лабораторной работе 7**

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

Тукаев Тимур Ильшатович НММбд-03-23

# Содержание

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

## Список иллюстраций

2.1	Программа в файле lab7-1.asm . . . . .	7
2.2	Запуск программы lab7-1.asm . . . . .	8
2.3	Программа в файле lab7-1.asm: . . . . .	9
2.4	Запуск программы lab7-1.asm: . . . . .	11
2.5	Программа в файле lab7-1.asm . . . . .	12
2.6	Запуск программы lab7-1.asm . . . . .	14
2.7	Программа в файле 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	Программа в файле task.asm . . . . .	21
2.13	Запуск программы task.asm . . . . .	24
2.14	Программа в файле task2.asm . . . . .	25
2.15	Запуск программы task2.asm . . . . .	27

## **Список таблиц**

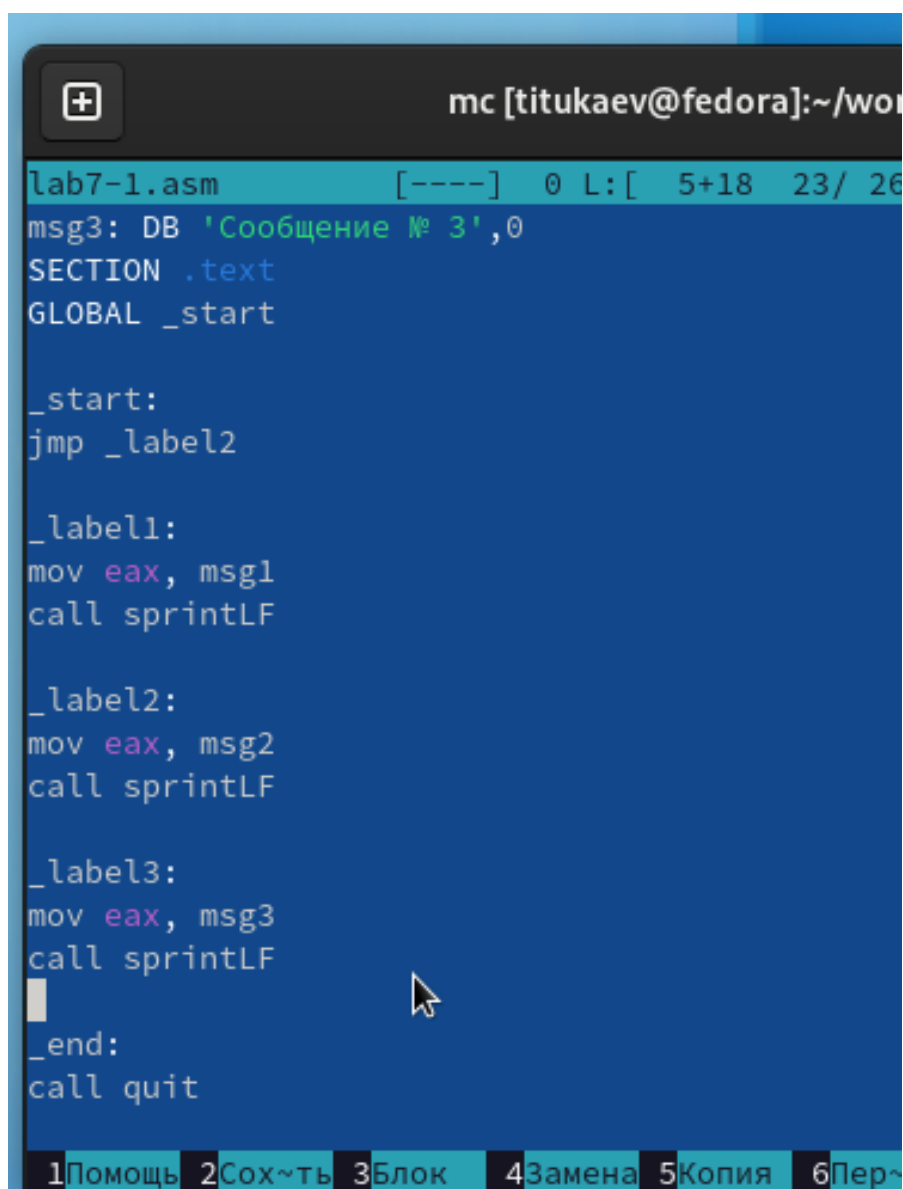
# 1 Цель работы

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. Создал каталог для программ 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.



```
lab7-1.asm [----] 0 L:[ 5+18 23/ 26
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
jmp _label2

_label1:
mov eax, msg1
call sprintfLF

_label2:
mov eax, msg2
call sprintfLF

_label3:
mov eax, msg3
call sprintfLF

_end:
call quit
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перезагрузить

Рис. 2.1: Программа в файле lab7-1.asm

Также размещаю код программы в отчете.

```
%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 sprintfLF

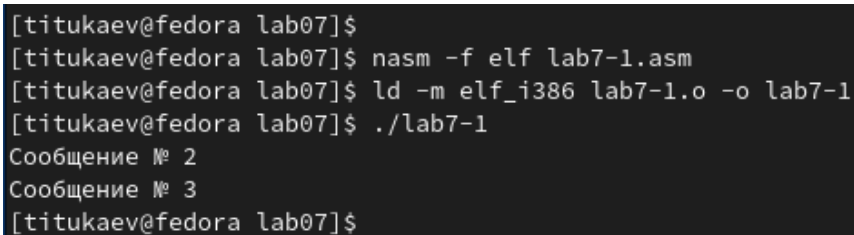
_label2:
mov eax, msg2
call sprintfLF

_label3:
mov eax, msg3
call sprintfLF

_end:
call quit

```

Создал исполняемый файл и запустил его.



```

[titukaev@fedora lab07]$
[titukaev@fedora lab07]$ nasm -f elf lab7-1.asm
[titukaev@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[titukaev@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[titukaev@fedora lab07]$

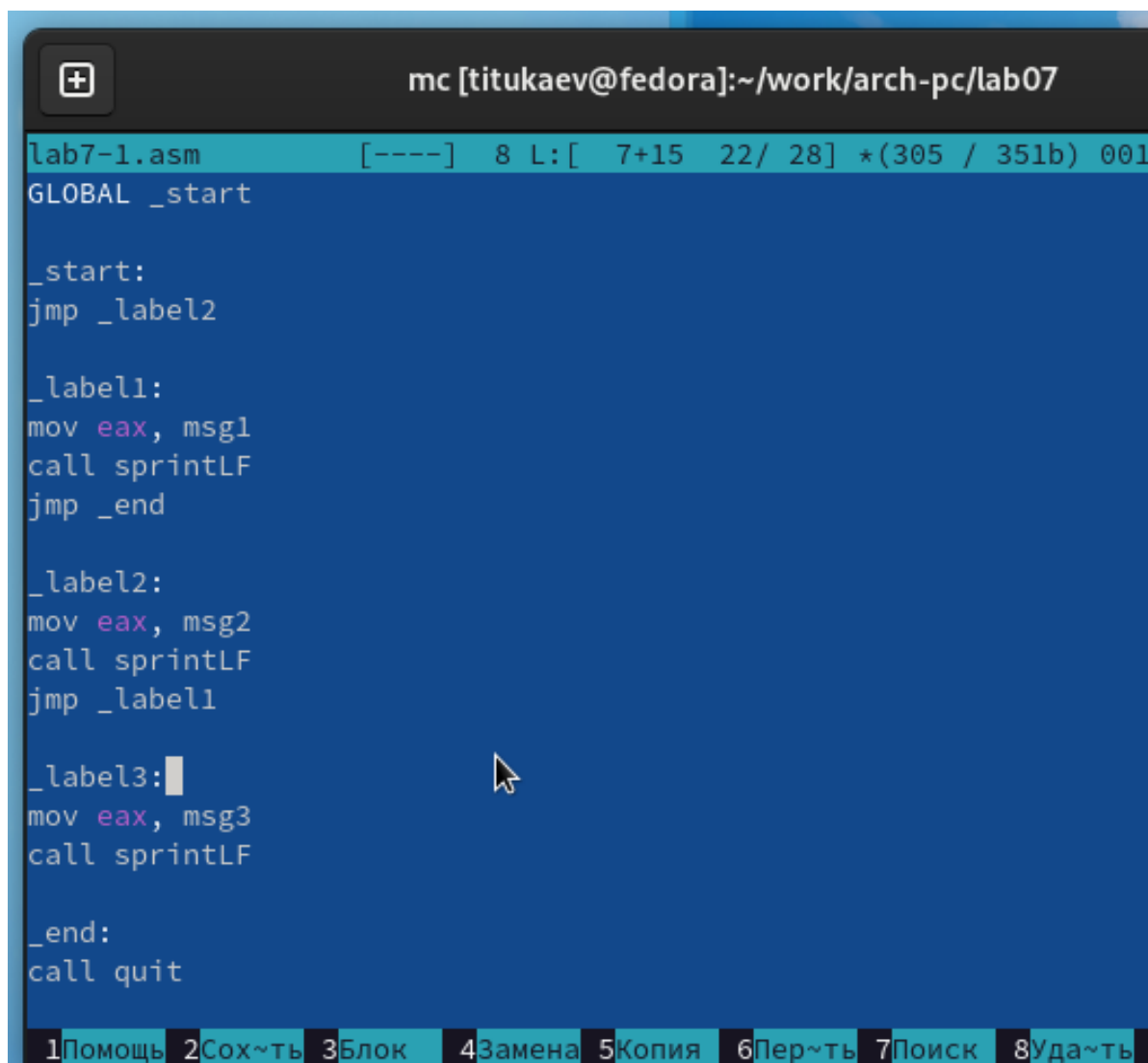
```

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.



```
mc [titukaev@fedora]:~/work/arch-pc/lab07
lab7-1.asm [----] 8 L:[ 7+15 22/ 28] *(305 / 351b) 001
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. 2.3: Программа в файле lab7-1.asm:

Также размещаю код программы в отчете.

```
%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 sprintfLF
jmp _end

_label2:
mov eax, msg2
call sprintfLF
jmp _label1

_label3:
mov eax, msg3
call sprintfLF

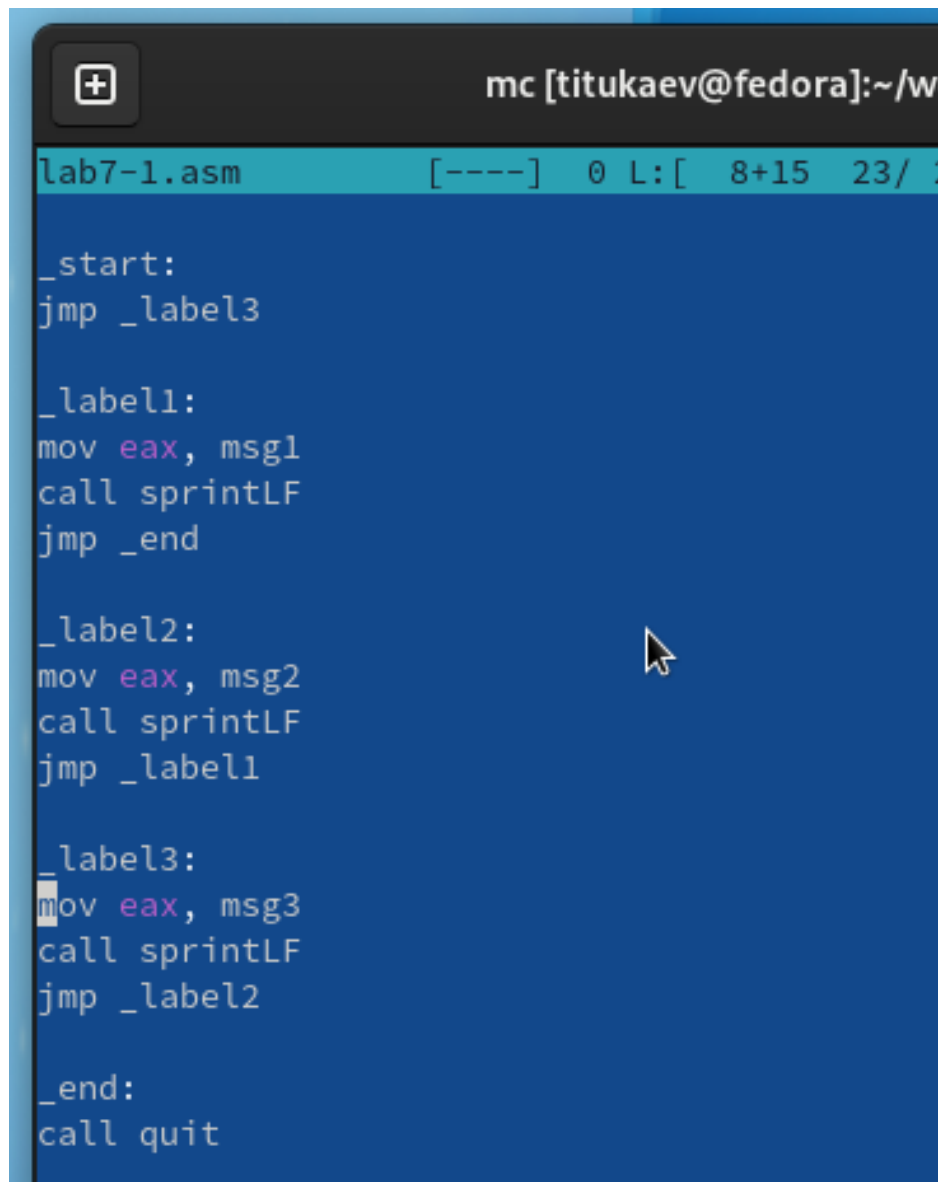
_end:
call quit
```

```
[titukaev@fedora lab07]$ nasm -f elf lab7-1.asm
[titukaev@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[titukaev@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 1
[titukaev@fedora 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



```
mc [titukaev@fedora]:~/w
lab7-1.asm [----] 0 L:[ 8+15 23/ 2
_start:
jmp _label3

_label1:
mov eax, msg1
call sprintf
jmp _end

_label2:
mov eax, msg2
call sprintf
jmp _label1

_label3:
mov eax, msg3
call sprintf
jmp _label2

_end:
call quit
```

Рис. 2.5: Программа в файле lab7-1.asm

Также размещаю код программы в отчете.

```
%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 sprintfLF
jmp _end
```

```
_label2:
mov eax, msg2
call sprintfLF
jmp _label1
```

```
_label3:
mov eax, msg3
call sprintfLF
jmp _label2
```

```
_end:
call quit
```

```
[titukaev@fedora lab07]$  
[titukaev@fedora lab07]$ nasm -f elf lab7-1.asm  
[titukaev@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1  
[titukaev@fedora lab07]$ ./lab7-1  
Сообщение № 3  
Сообщение № 2  
Сообщение № 1  
[titukaev@fedora 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.

```
mc [titukaev@fedora]:~/work/arch-pc/lab07
lab7-2.asm [----] 0 L:[ 12+ 7 19/ 50] *(342 /1057b) 0099
_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
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. 2.7: Программа в файле lab7-2.asm

Также размещаю код программы в отчете.

```
%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
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

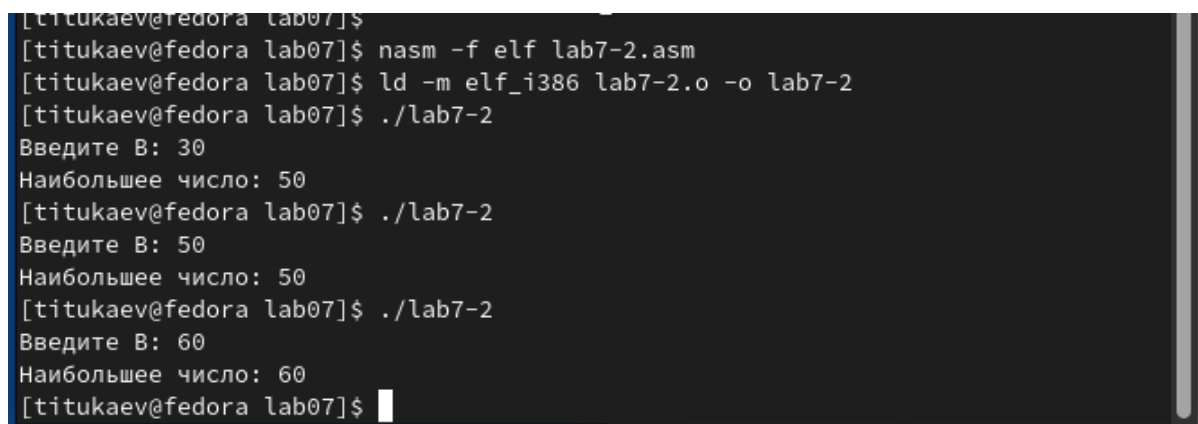
```



```

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

```



```

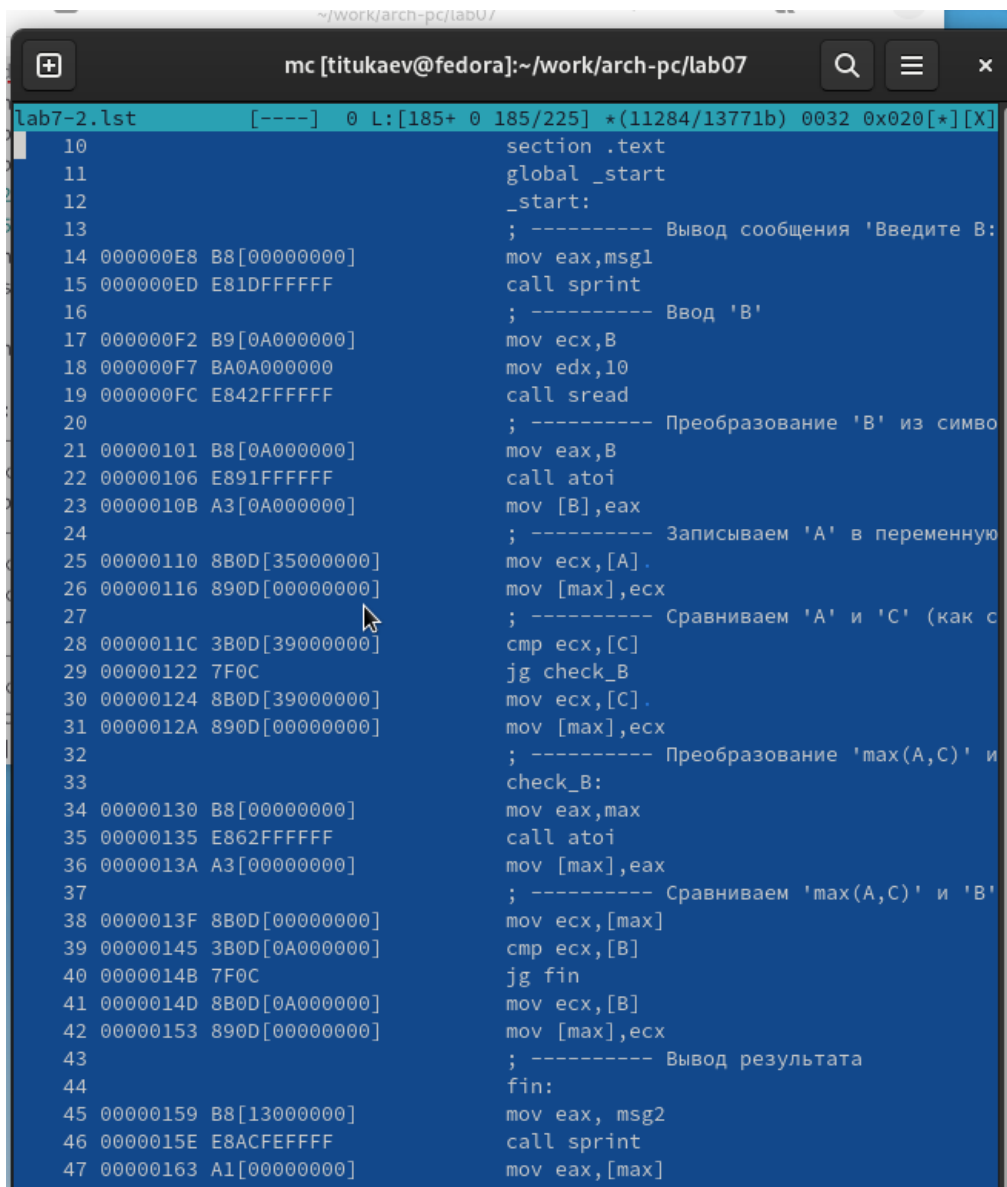
[titukaev@fedora lab07]$
[titukaev@fedora lab07]$ nasm -f elf lab7-2.asm
[titukaev@fedora lab07]$ ld -m elf_i386 lab7-2.o -o lab7-2
[titukaev@fedora lab07]$ ./lab7-2
Введите B: 30
Наибольшее число: 50
[titukaev@fedora lab07]$ ./lab7-2
Введите B: 50
Наибольшее число: 50
[titukaev@fedora lab07]$ ./lab7-2
Введите B: 60
Наибольшее число: 60
[titukaev@fedora 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 [-----] 0 L:[185+ 0 185/225] *(11284/13771b) 0032 0x020[*][X]
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B:
14 000000E8 B8[00000000] mov eax,msg1
15 000000ED E81DFFFFFF call sprint
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 симво
21 00000101 B8[0A000000] mov eax,B
22 00000106 E891FFFFFF call atoi
23 0000010B A3[0A000000] mov [B],eax
24 ; ----- Записываем 'A' в переменную
25 00000110 8B0D[35000000] mov ecx,[A]
26 00000116 890D[00000000] mov [max],ecx
27 ; ----- Сравниваем 'A' и 'C' (как с
28 0000011C 3B0D[39000000] cmp ecx,[C]
29 00000122 7F0C jg check_B
30 00000124 8B0D[39000000] mov ecx,[C]
31 0000012A 890D[00000000] mov [max],ecx
32 ; ----- Преобразование 'max(A,C)' и
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'
38 0000013F 8B0D[00000000] mov ecx,[max]
39 00000145 3B0D[0A000000] cmp ecx,[B]
40 0000014B 7F0C jg fin
41 0000014D 8B0D[0A000000] mov ecx,[B]
42 00000153 890D[00000000] mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 00000159 B8[13000000] mov eax,msg2
46 0000015E E8ACFEFFFF call sprint
47 00000163 A1[00000000] mov eax,[max]
```

Рис. 2.9: Файл листинга lab7-2

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

строка 21

- 21 - номер строки

- 00000101 - адрес
- B8[0A000000] - машинный код
- mov eax,B - код программы

строка 22

- 22 - номер строки
- 00000106 - адрес
- E891FFFFFF - машинный код
- call atoi- код программы

строка 23

- 23 - номер строки
- 0000010B - адрес
- A3[0A000000] - машинный код
- mov [B],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.

```
[titukaev@fedora lab07]$  
[titukaev@fedora lab07]$ nasm -f elf lab7-2.asm -l lab7-2.lst  
lab7-2.asm:17: error: invalid combination of opcode and operands  
[titukaev@fedora lab07]$
```

Рис. 2.10: Ошибка трансляции lab7-2

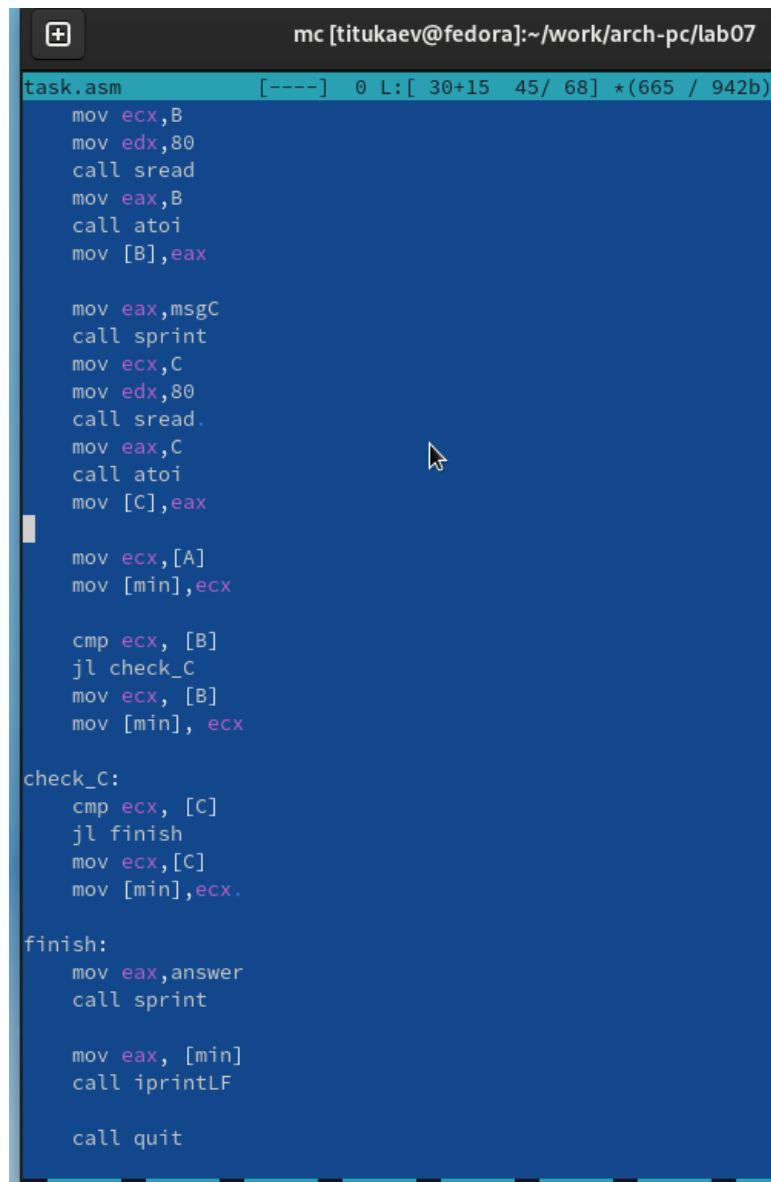
```
lab7-2.lst [----] 22 L:[181+18 199/226] *(12199/13861b) 0048 0x030[*] [X]
6 00000039 35300000 C dd '50'
7 section .bss
8 00000000 <res Ah> max resb 10
9 0000000A <res Ah> B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B:
14 000000E8 B8[00000000] mov eax,msg1
15 000000ED E81DFFFFFF call sprint
16 ; ----- Ввод 'B'
17 mov ecx,
17 ***** error: invalid combination of opcode and operand
18 000000F2 BA0A000000 mov edx,10
19 000000F7 E847FFFFFF call sread
20 ; ----- Преобразование 'B' из символа
21 000000FC B8[0A000000] mov eax,B
22 00000101 E896FFFFFF call atoi
23 00000106 A3[0A000000] mov [B],eax
24 ; ----- Записываем 'A' в переменную
25 0000010B 8B0D[35000000] mov ecx,[A]
26 00000111 890D[00000000] mov [max],ecx
```

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 5 - 54, 62, 87



```
task.asm [----] 0 L:[ 30+15 45/ 68] *(665 / 942b)
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

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

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

check_C:
cmp ecx,[C]
jnl finish
mov ecx,[C]
mov [min],ecx.

finish:
mov eax,answer
call sprint

mov eax,[min]
call iprintLF

call quit
```

Рис. 2.12: Программа в файле task.asm

Также размещаю код программы в отчете.

```
%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
```

```
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
```

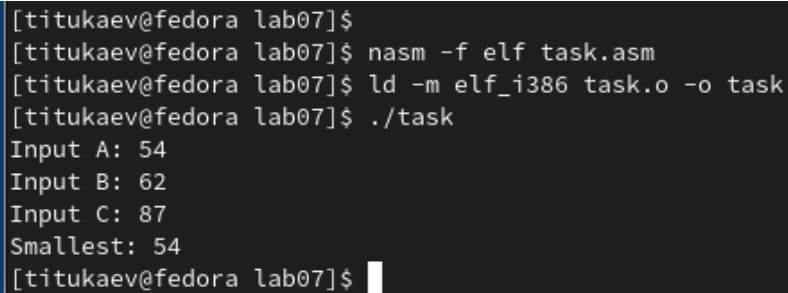
```
finish:
```

```
mov eax,answer  
call sprint
```

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

```
call iprintLF
```

```
call quit
```



```
[titukaev@fedora lab07]$  
[titukaev@fedora lab07]$ nasm -f elf task.asm  
[titukaev@fedora lab07]$ ld -m elf_i386 task.o -o task  
[titukaev@fedora lab07]$ ./task  
Input A: 54  
Input B: 62  
Input C: 87  
Smallest: 54  
[titukaev@fedora lab07]$
```

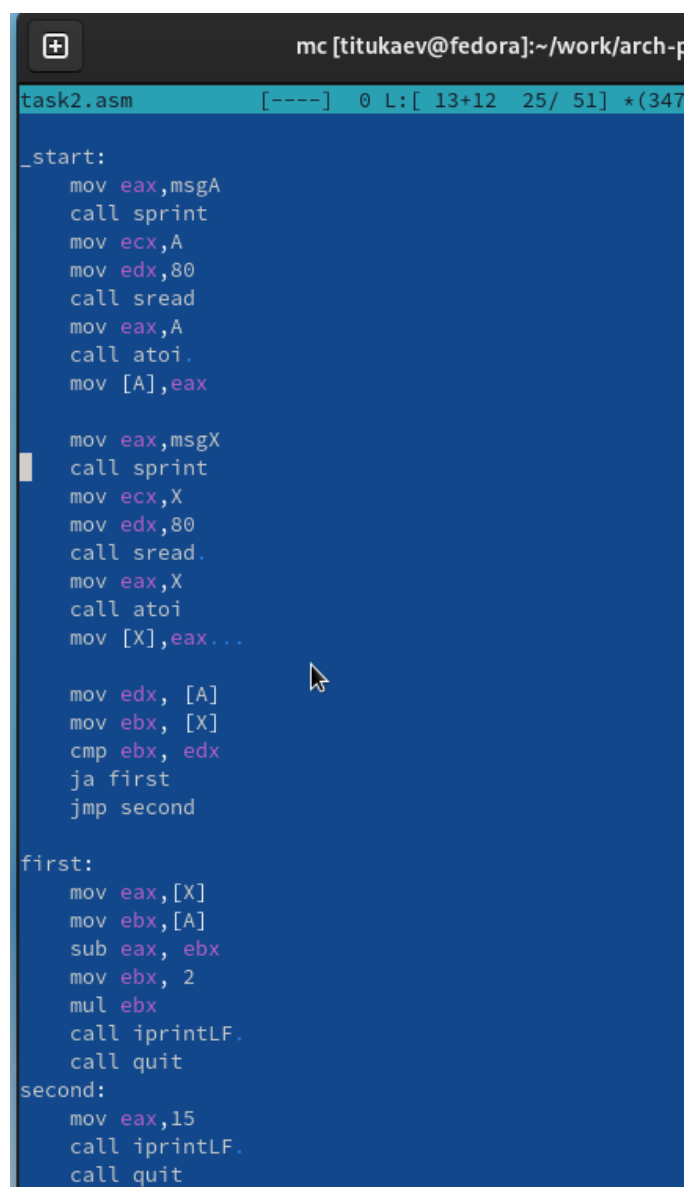
Рис. 2.13: Запуск программы task.asm

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 5

$$\begin{cases} 2(x - a), x > a \\ 15, x \leq a \end{cases}$$





```
mc [titukaev@fedora]:~/work/arch-p
task2.asm [----] 0 L: [ 13+12 25/ 51] *(347)

_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...

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

first:
    mov eax,[X]
    mov ebx,[A]
    sub eax, ebx
    mov ebx, 2
    mul ebx
    call iprintLF.
    call quit

second:
    mov eax,15
    call iprintLF.
    call quit
```

Рис. 2.14: Программа в файле task2.asm

Также размещаю код программы в отчете.

```
%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

    mov edx, [A]
    mov ebx, [X]
```

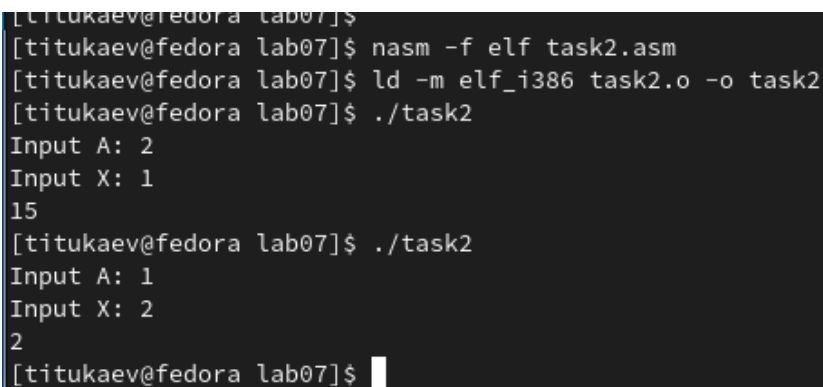
```
    cmp ebx, edx
    ja first
    jmp second
```

first:

```
    mov eax,[X]
    mov ebx,[A]
    sub eax, ebx
    mov ebx, 2
    mul ebx
    call iprintLF
    call quit
```

second:

```
    mov eax,15
    call iprintLF
    call quit
```



```
[titukaev@fedora lab07]$  
[titukaev@fedora lab07]$ nasm -f elf task2.asm  
[titukaev@fedora lab07]$ ld -m elf_i386 task2.o -o task2  
[titukaev@fedora lab07]$ ./task2  
Input A: 2  
Input X: 1  
15  
[titukaev@fedora lab07]$ ./task2  
Input A: 1  
Input X: 2  
2  
[titukaev@fedora lab07]$
```

Рис. 2.15: Запуск программы task2.asm

## 3 Выводы

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