

Отчёта по лабораторной работе 8

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

Еюбоглу Тимур НПИбд-01-22

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	24
	Список литературы	25

Список иллюстраций

4.1	Файл lab8-1.asm:	9
4.2	Программа lab8-1.asm:	10
4.3	Файл lab8-1.asm:	11
4.4	Программа lab8-1.asm:	12
4.5	Файл lab8-1.asm	13
4.6	Программа lab8-1.asm	14
4.7	Файл lab8-2.asm	15
4.8	Программа lab8-2.asm	16
4.9	Файл листинга lab8-2	17
4.10	ошибка трансляции lab8-2	18
4.11	файл листинга с ошибкой lab8-2	19
4.12	Файл lab8-3.asm	20
4.13	Программа lab8-3.asm	21
4.14	Файл lab8-4.asm	22
4.15	Программа lab8-4.asm	23

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 Задание

1. Изучите примеры программ.
2. Изучите файл листинга.
3. Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c . Значения переменных выбрать из табл. 8.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу
4. 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 8.6 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 8.6.

3 Теоретическое введение

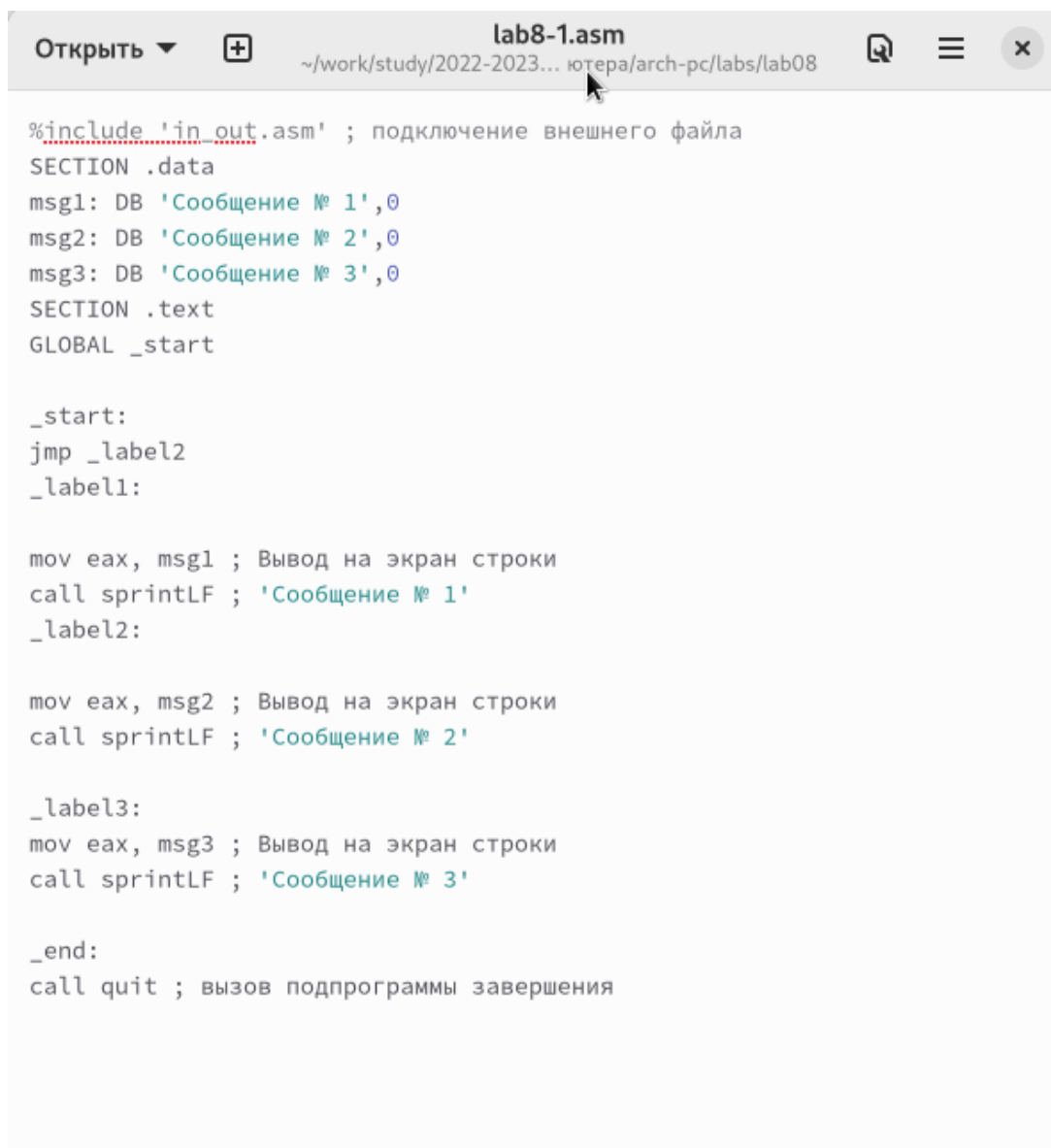
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

4 Выполнение лабораторной работы

1. Создайте каталог для программ лабораторной работы № 8, перейдите в него и создайте файл `lab8-1.asm`
2. Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Введите в файл `lab8-1.asm` текст программы из листинга 8.1. (рис. 4.1)



```
%include 'in_out.asm' ; подключение внешнего файла
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 sprintfLF ; 'Сообщение № 1'
_label2:

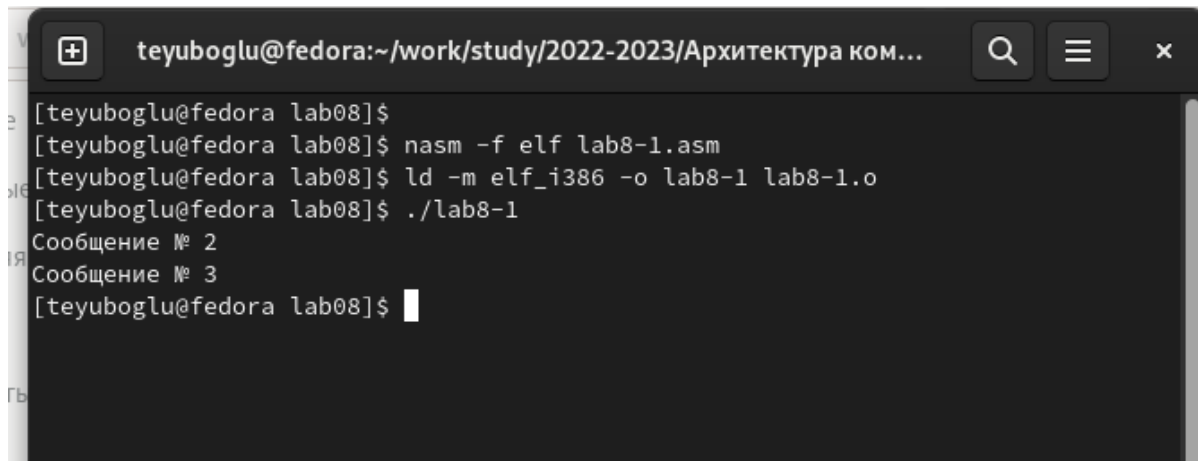
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 2'

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

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.1: Файл lab8-1.asm:

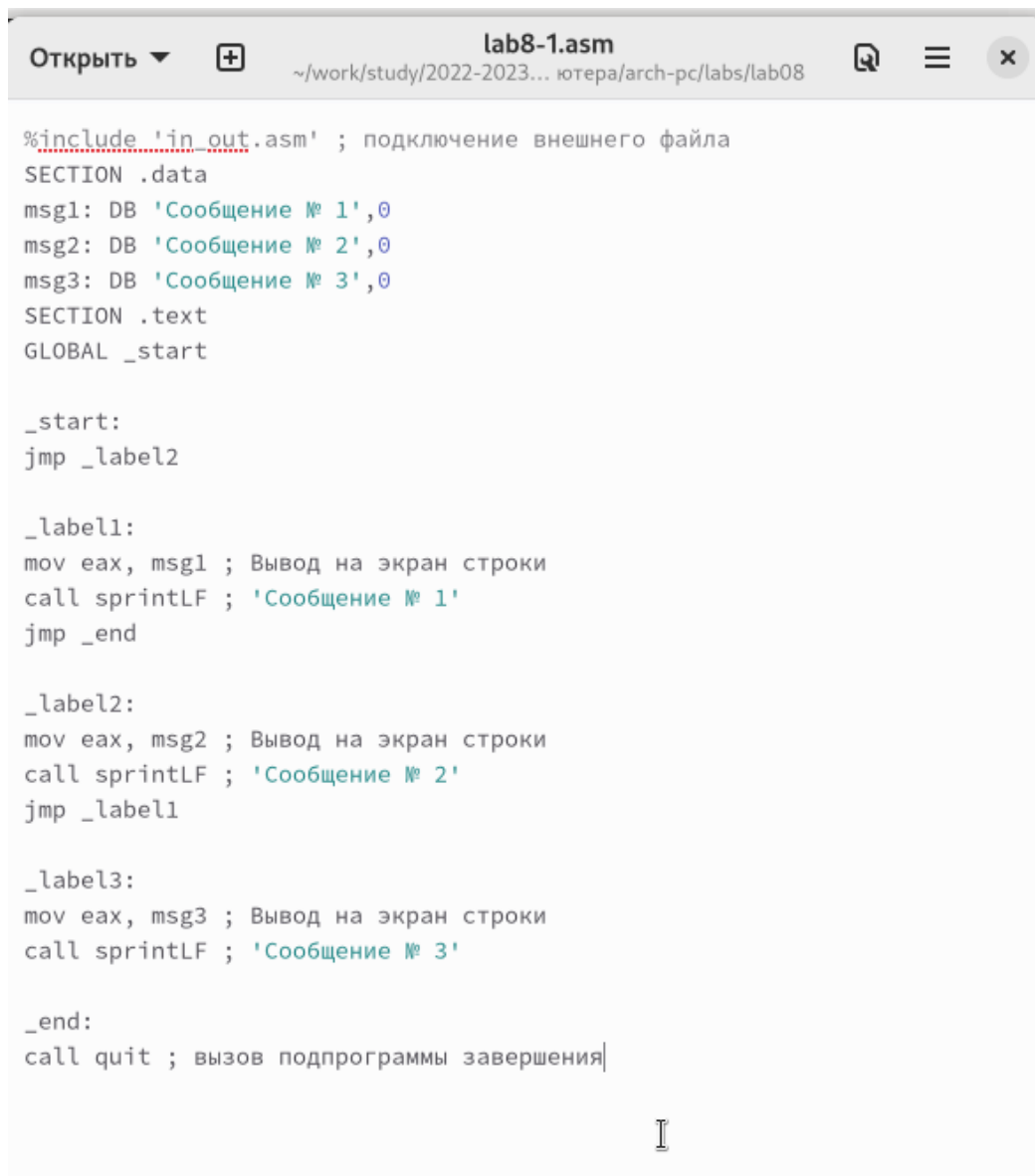
Создайте исполняемый файл и запустите его. (рис. 4.2)

A terminal window with a dark background and light text. The title bar shows the user 'teyuboglu@fedora' and the path '~/work/study/2022-2023/Архитектура ком...'. The terminal contains the following commands and output:

```
[teyuboglu@fedora lab08]$  
[teyuboglu@fedora lab08]$ nasm -f elf lab8-1.asm  
[teyuboglu@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o  
[teyuboglu@fedora lab08]$ ./lab8-1  
Сообщение № 2  
Сообщение № 3  
[teyuboglu@fedora lab08]$
```

Рис. 4.2: Программа lab8-1.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 8.2. (рис. 4.3, 4.4)



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

_start:
jmp _label2

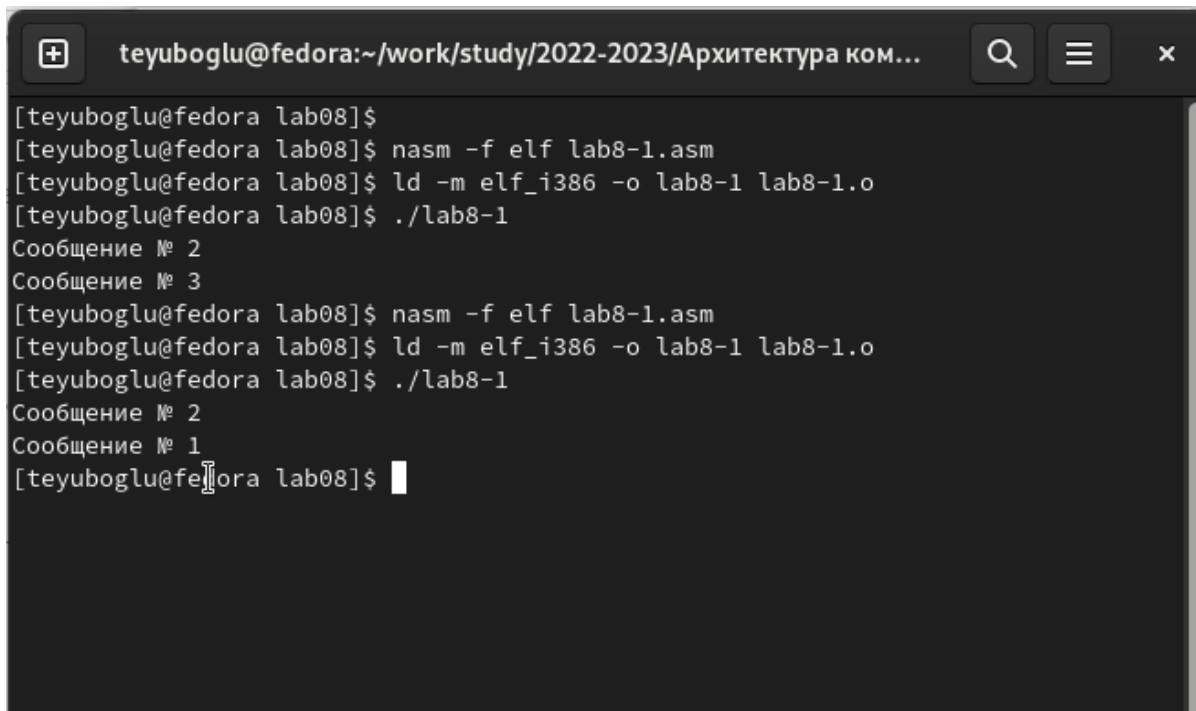
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
jmp _end

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

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

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.3: Файл lab8-1.asm:

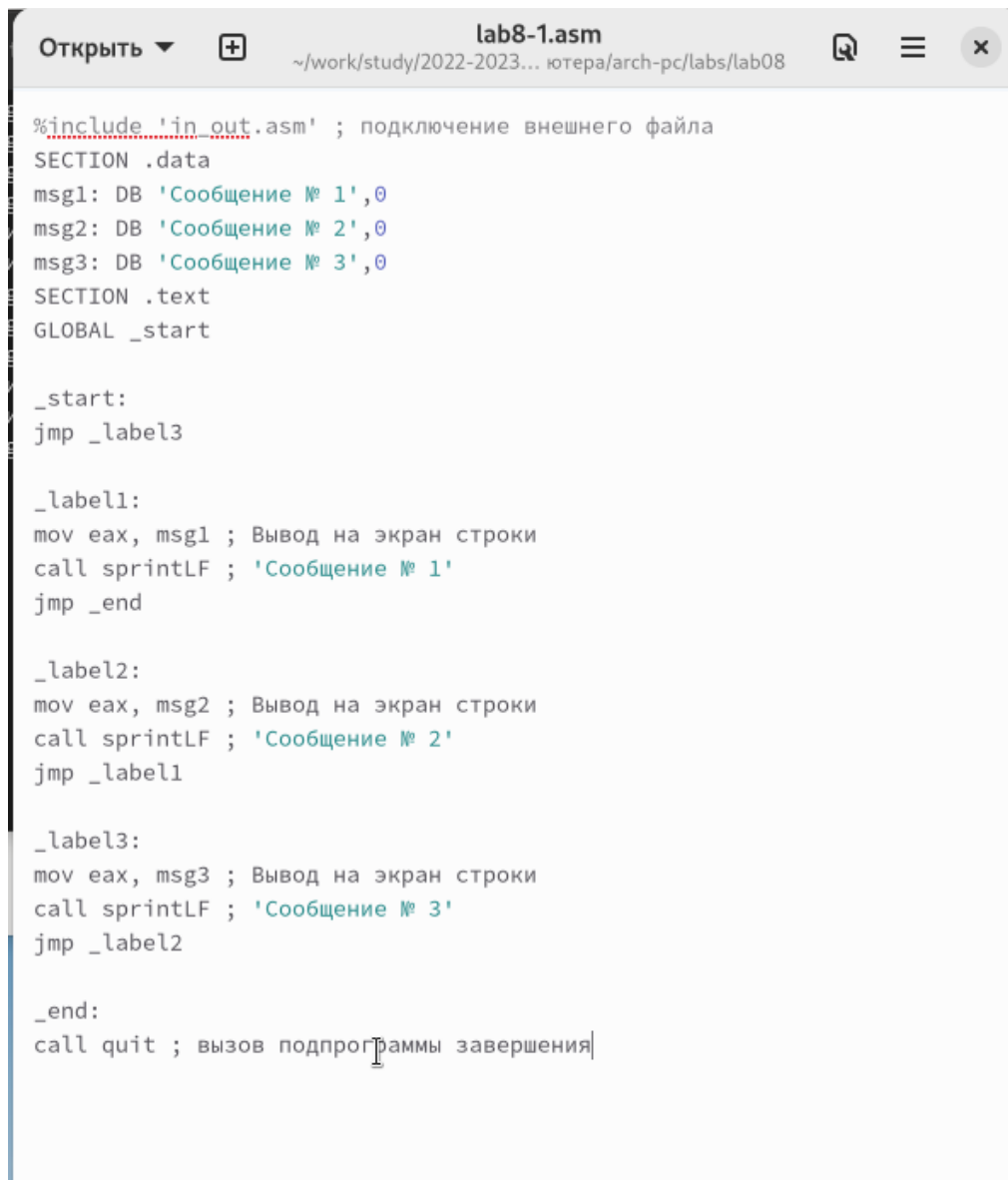
A terminal window with a dark background and light text. The title bar shows the user 'teyuboglu@fedora' and the path '~/work/study/2022-2023/Архитектура ком...'. The terminal contains the following text:

```
[teyuboglu@fedora lab08]$  
[teyuboglu@fedora lab08]$ nasm -f elf lab8-1.asm  
[teyuboglu@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o  
[teyuboglu@fedora lab08]$ ./lab8-1  
Сообщение № 2  
Сообщение № 3  
[teyuboglu@fedora lab08]$ nasm -f elf lab8-1.asm  
[teyuboglu@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o  
[teyuboglu@fedora lab08]$ ./lab8-1  
Сообщение № 2  
Сообщение № 1  
[teyuboglu@fedora lab08]$
```

Рис. 4.4: Программа lab8-1.asm:

Измените текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим (рис. 4.5, 4.6):

Сообщение № 3
Сообщение № 2
Сообщение № 1



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

_start:
jmp _label3

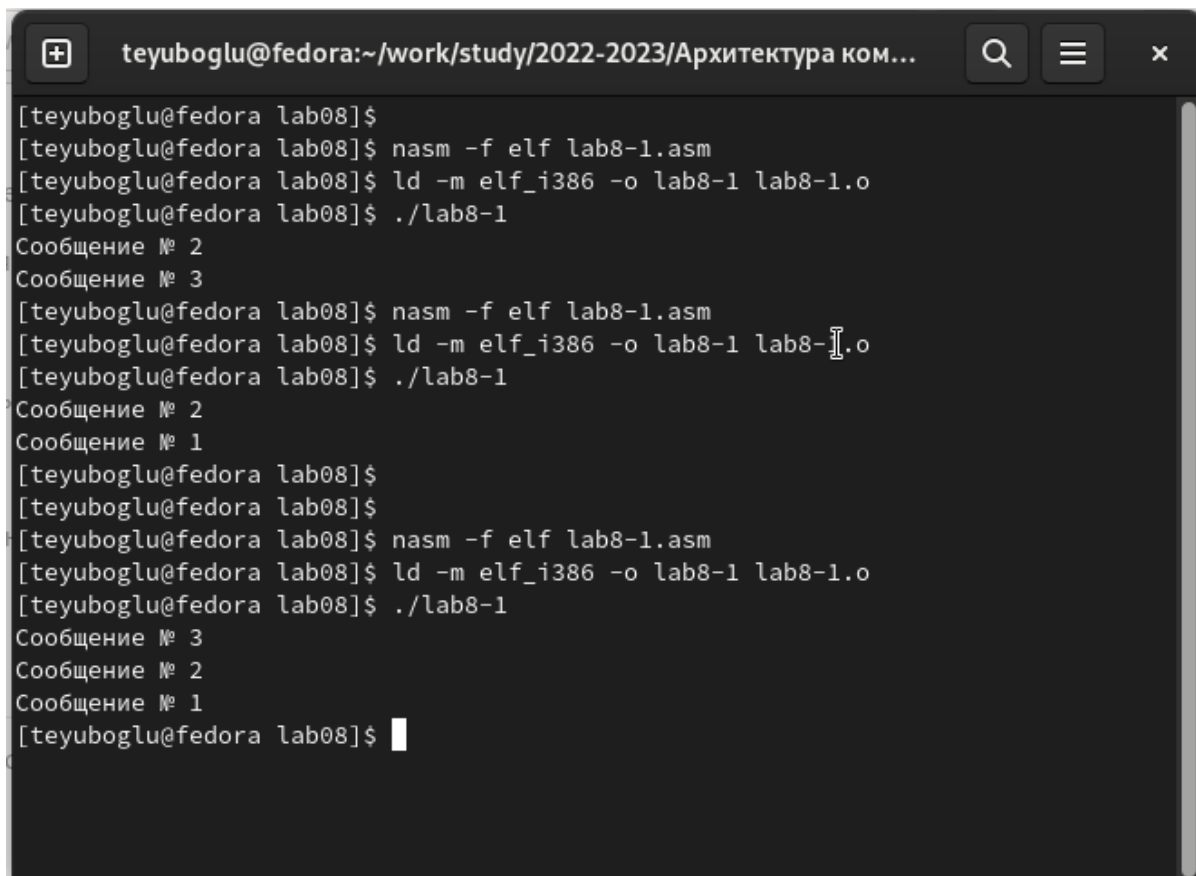
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end

_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
```

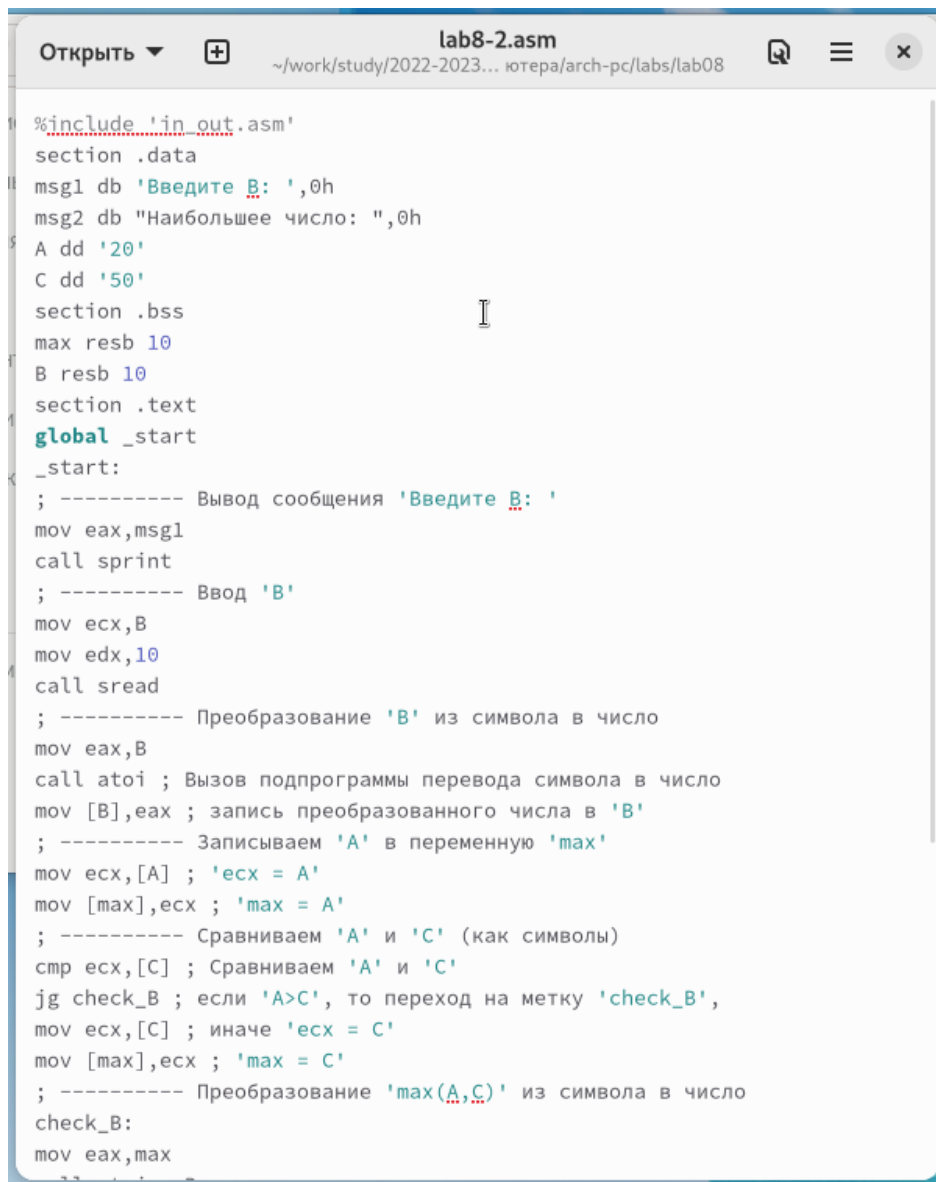
Рис. 4.5: Файл lab8-1.asm



```
teyuboglu@fedora:~/work/study/2022-2023/Архитектура ком...
[teyuboglu@fedora lab08]$
[teyuboglu@fedora lab08]$ nasm -f elf lab8-1.asm
[teyuboglu@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[teyuboglu@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[teyuboglu@fedora lab08]$ nasm -f elf lab8-1.asm
[teyuboglu@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[teyuboglu@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
[teyuboglu@fedora lab08]$
[teyuboglu@fedora lab08]$
[teyuboglu@fedora lab08]$ nasm -f elf lab8-1.asm
[teyuboglu@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[teyuboglu@fedora lab08]$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[teyuboglu@fedora lab08]$
```

Рис. 4.6: Программа lab8-1.asm

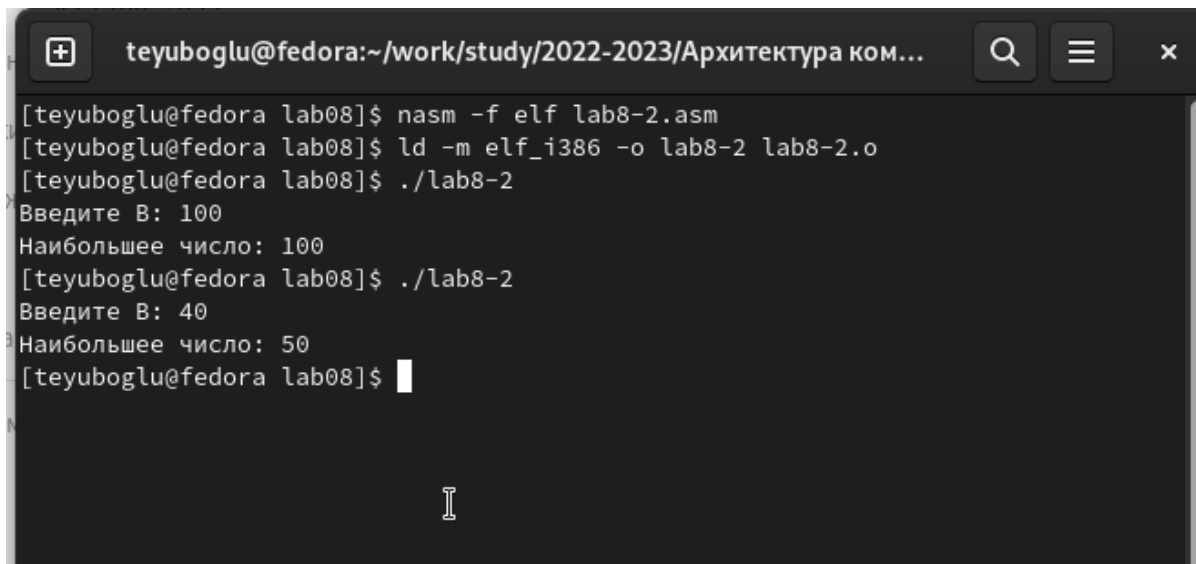
- Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры. Создайте исполняемый файл и проверьте его работу для разных значений В. (рис. 4.7, 4.8)



```
lab8-2.asm
~/work/study/2022-2023... ютеpa/arch-pc/labs/lab08

%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
```

Рис. 4.7: Файл lab8-2.asm



```
teyuboglu@fedora:~/work/study/2022-2023/Архитектура ком...
[teyuboglu@fedora lab08]$ nasm -f elf lab8-2.asm
[teyuboglu@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[teyuboglu@fedora lab08]$ ./lab8-2
Введите B: 100
Наибольшее число: 100
[teyuboglu@fedora lab08]$ ./lab8-2
Введите B: 40
Наибольшее число: 50
[teyuboglu@fedora lab08]$
```

Рис. 4.8: Программа lab8-2.asm

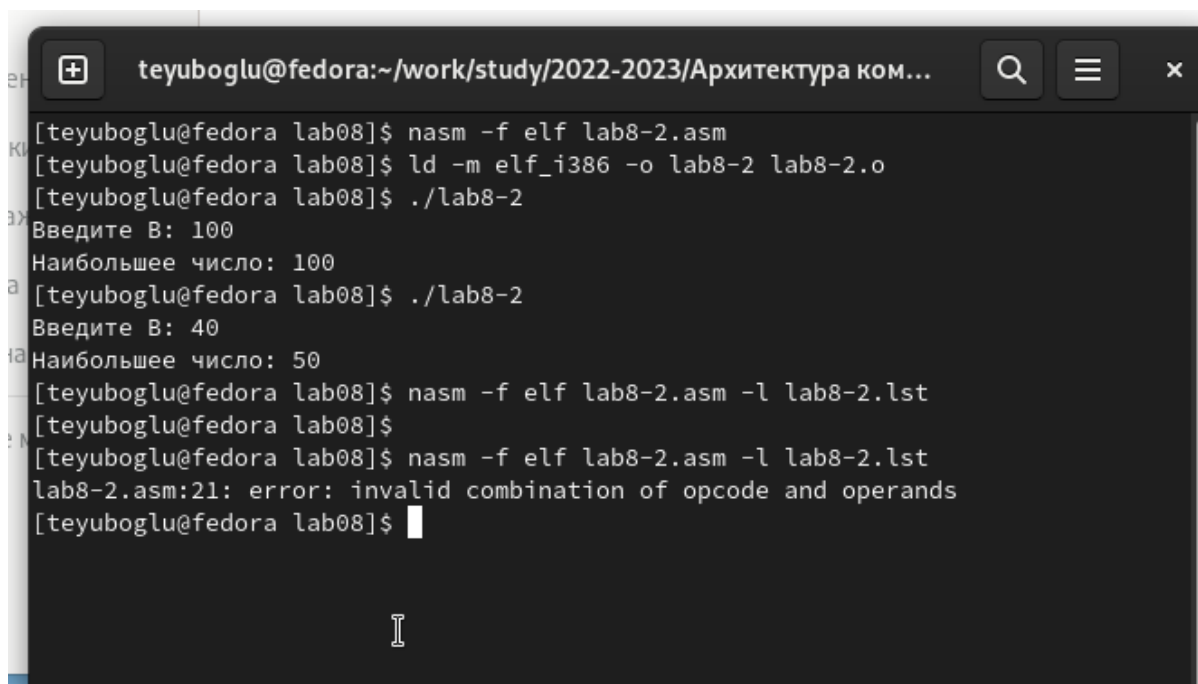
4. Обычно `nasm` создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ `-l` и задав имя файла листинга в командной строке. Создайте файл листинга для программы из файла `lab8-2.asm` (рис. 4.9)

- 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

Откройте файл с программой lab8-2.asm и в любой инструкции с двумя операндами удалить один операнд. Выполните трансляцию с получением файла листинга (рис. 4.10,4.11)



```

teyuboglu@fedora:~/work/study/2022-2023/Архитектура ком...
[teyuboglu@fedora lab08]$ nasm -f elf lab8-2.asm
[teyuboglu@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[teyuboglu@fedora lab08]$ ./lab8-2
Введите B: 100
Наибольшее число: 100
[teyuboglu@fedora lab08]$ ./lab8-2
Введите B: 40
Наибольшее число: 50
[teyuboglu@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst
[teyuboglu@fedora lab08]$
[teyuboglu@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst
lab8-2.asm:21: error: invalid combination of opcode and operands
[teyuboglu@fedora lab08]$

```

Рис. 4.10: ошибка трансляции lab8-2

```

6 00000039 35300000      C dd '50'
7                                section .bss
8 00000000 <res Ah>    max resb 10
9 00000000 <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 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                                mov eax,
22                                *****
23                                error: invalid combination of opcode and operands
24                                call atoi ; Вызов подпрограммы перевода символа в число
25 00000106 A3[0A000000]    mov [B],eax ; запись преобразованного числа в 'B'
26                                ; ----- Записываем 'A' в переменную 'max'
27 0000010B 8B0D[35000000]    mov ecx,[A] ; 'ecx' = A
28 00000111 890D[00000000]    mov [max],ecx ; 'max' = A
29                                ; ----- Сравниваем 'A' и 'C' (как символы)
30 00000117 3B0D[39000000]    cmp ecx,[C] ; Сравниваем 'A' и 'C'
31 0000011D 7F0C            jg check_B ; если 'A>C', то переход на метку 'check_B',
32 0000011F 8B0D[39000000]    mov ecx,[C] ; иначе 'ecx' = C
33 00000125 890D[00000000]    mov [max],ecx ; 'max' = C
34                                ; ----- Преобразование 'max(A,C)' из символа в число
35                                check_B:
36 0000012B B8[00000000]    mov eax,max
37 00000130 E867FFFFFF    call atoi ; Вызов подпрограммы перевода символа в число
38 00000135 A3[00000000]    mov [max],eax ; запись преобразованного числа в 'max'
39                                ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)

```

Рис. 4.11: файл листинга с ошибкой lab8-2

5. Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c. Значения переменных выбрать из табл. 8.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу (рис. 4.12,4.13)

для варианта 18 - 83,73,30

```
Открыть ▾ + lab8-3.asm
~/.work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08

%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

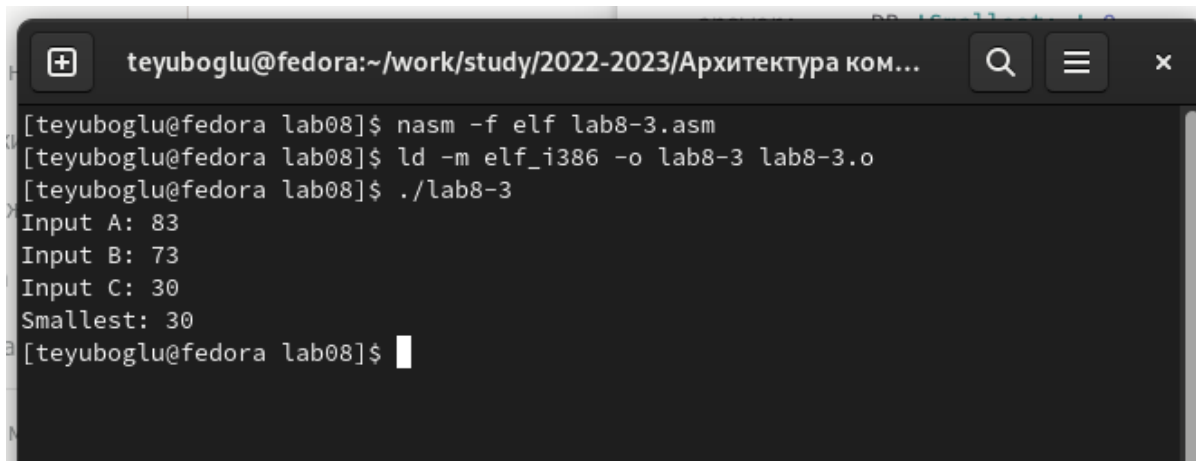
SECTION .bss
    A: RESB 80
    B: RESB 80
    C: RESB 80
    result:    RESB 80
    min: RESB 80

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
```

Рис. 4.12: Файл lab8-3.asm



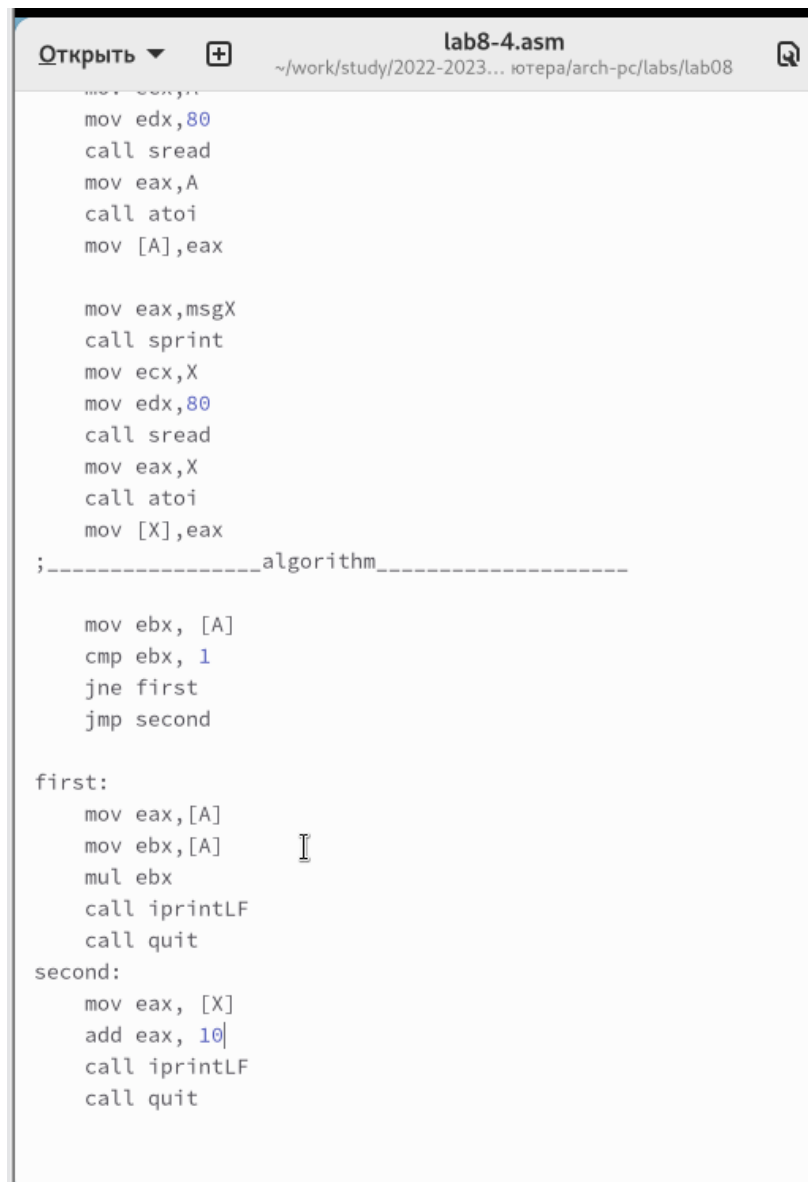
```
teyuboglu@fedora:~/work/study/2022-2023/Архитектура ком...
[teyuboglu@fedora lab08]$ nasm -f elf lab8-3.asm
[teyuboglu@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[teyuboglu@fedora lab08]$ ./lab8-3
Input A: 83
Input B: 73
Input C: 30
Smallest: 30
[teyuboglu@fedora lab08]$
```

Рис. 4.13: Программа lab8-3.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 8.6 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 8.6. (рис. 4.14,4.15)

для варианта 18

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



```
lab8-4.asm
~/work/study/2022-2023... ютеpa/arch-pc/labs/lab08

mov ecx, 1
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

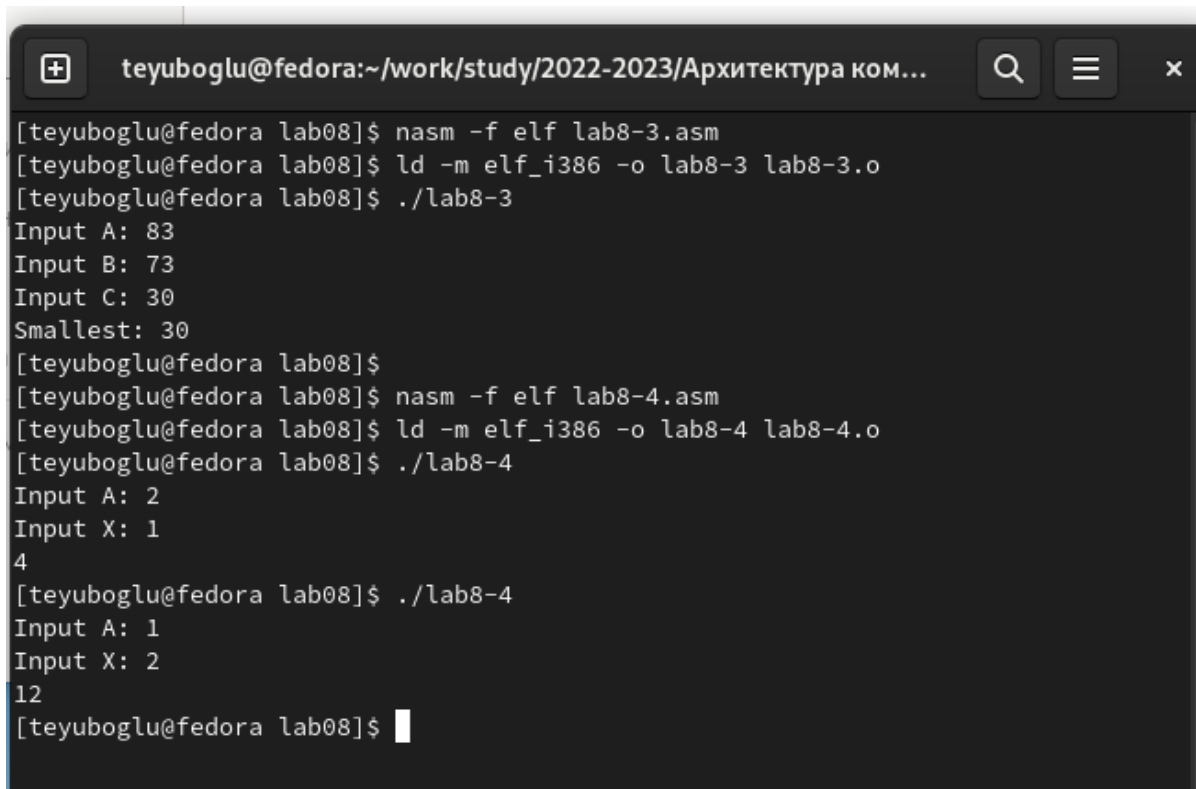
;-----algorithm-----

mov ebx, [A]
cmp ebx, 1
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
```

Рис. 4.14: Файл lab8-4.asm



```
teyuboglu@fedora:~/work/study/2022-2023/Архитектура ком...
[teyuboglu@fedora lab08]$ nasm -f elf lab8-3.asm
[teyuboglu@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[teyuboglu@fedora lab08]$ ./lab8-3
Input A: 83
Input B: 73
Input C: 30
Smallest: 30
[teyuboglu@fedora lab08]$
[teyuboglu@fedora lab08]$ nasm -f elf lab8-4.asm
[teyuboglu@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o
[teyuboglu@fedora lab08]$ ./lab8-4
Input A: 2
Input X: 1
4
[teyuboglu@fedora lab08]$ ./lab8-4
Input A: 1
Input X: 2
12
[teyuboglu@fedora lab08]$
```

Рис. 4.15: Программа lab8-4.asm

5 Выводы

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

Список литературы

1. Расширенный ассемблер: NASM
2. MASM, TASM, FASM, NASM под Windows и Linux