Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Бережной Иван Александрович

Содержание

# 1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

# 2 Задание

1. Разбор основ по переходам в NASM
2. Изучение структуры файла листинга
3. Задание для самостоятельной работы

# 3 Выполнение лабораторной работы

## 3.1 Разбор основ по переходам в NASM

Создадим каталог для дальнейшей работы, а также файл в нём lab7-1.asm (рис. [1](#fig:001)). Скопируем в новый файл код из предложенного листинга (рис. [2](#fig:002)), затем создадим и запустим исполняемый файл (рис. [3](#fig:003)).

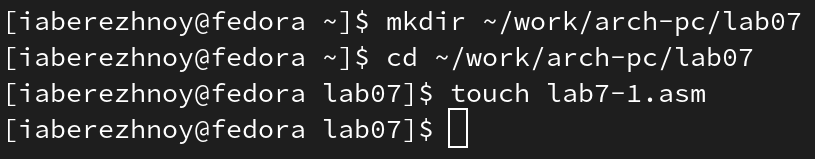


Figure 1: Создание файла lab7-1.asm

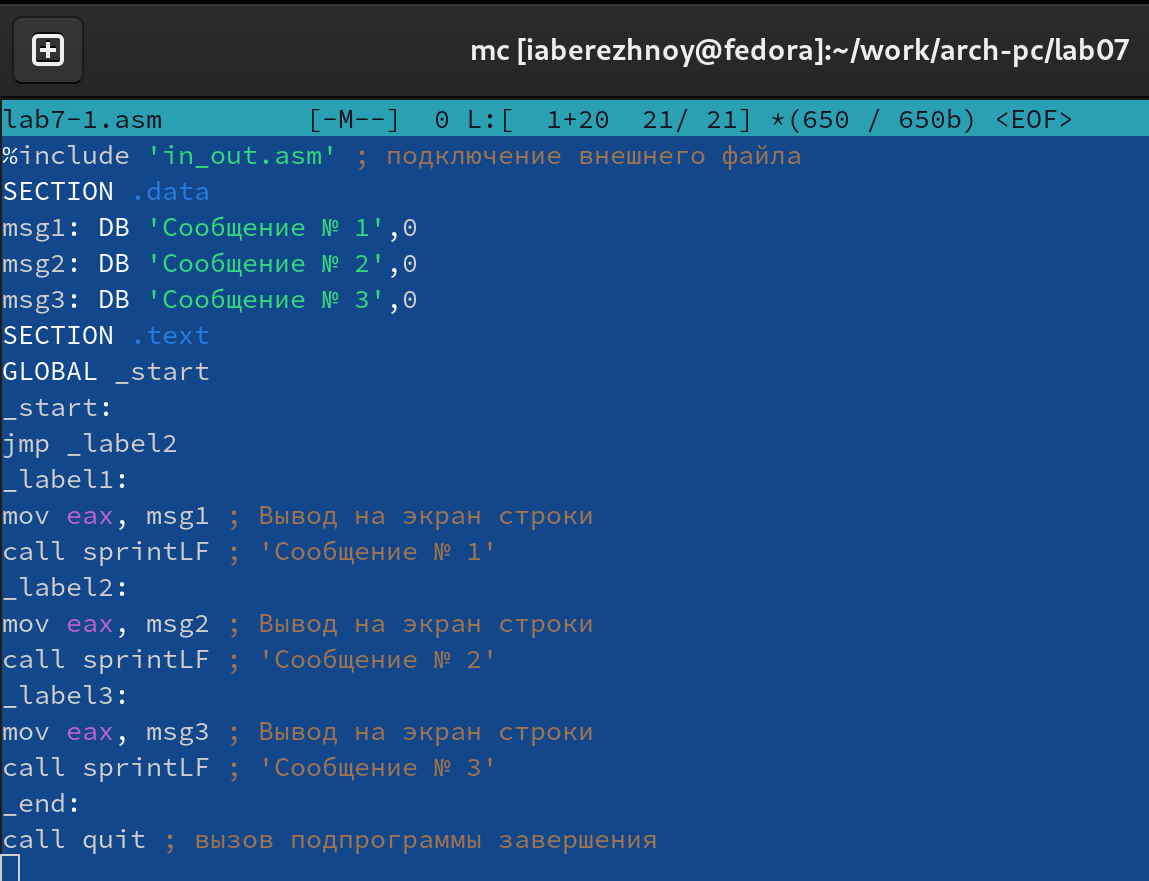


Figure 2: Копирование кода в файл lab7-1.asm

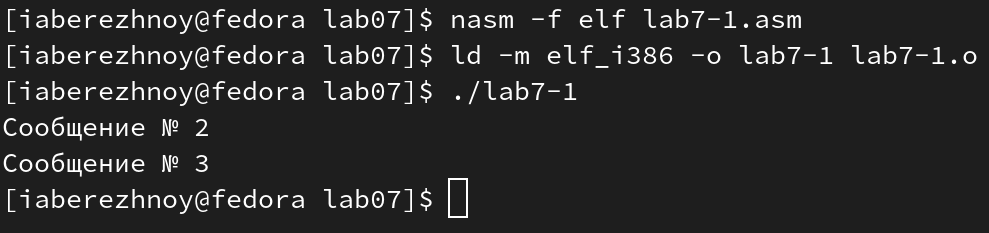


Figure 3: Запуск исполняемого файла lab7-1

Немного изменим программу, добавив инструкцию jmp\_label1 после вывода сообщения №2 (рис. [4](#fig:004)). Повторно создадим и запустим исполняемый файл рис. [5](#fig:005)).

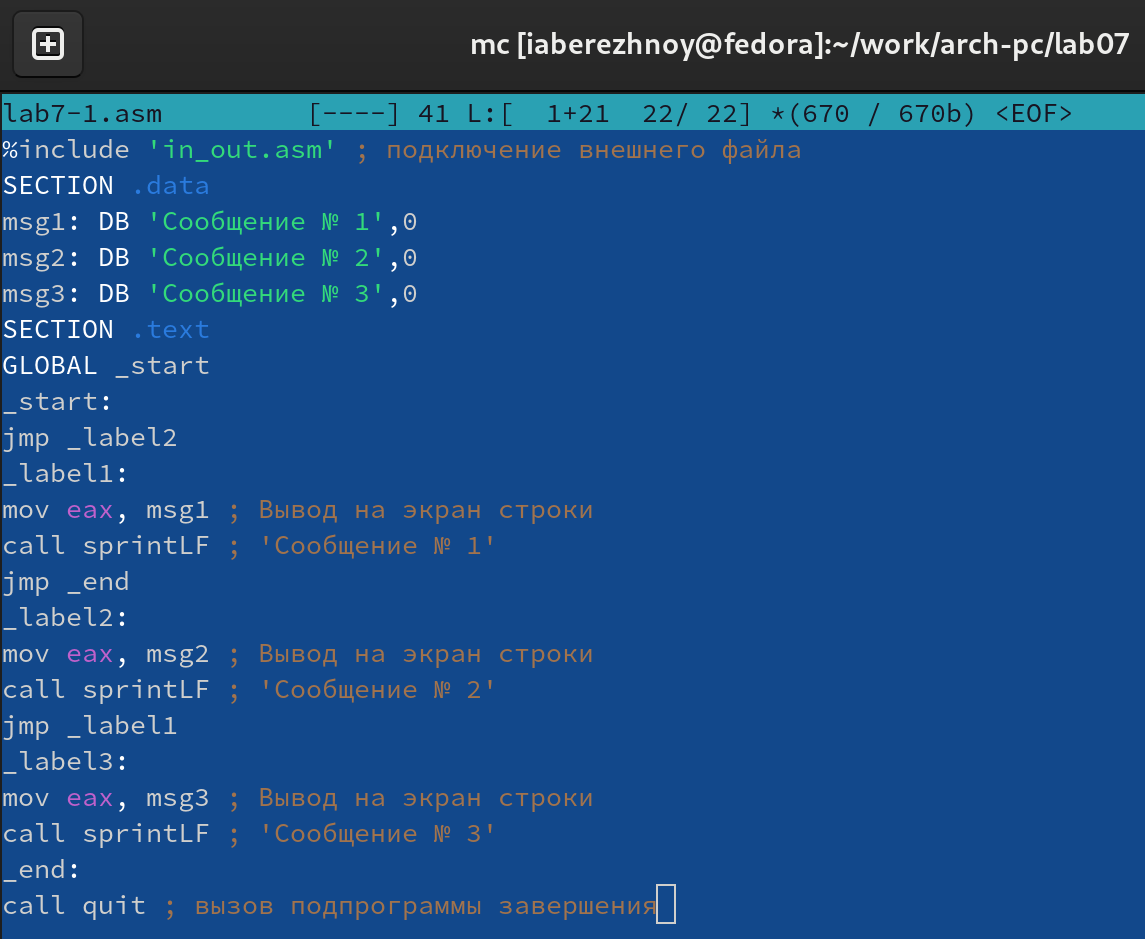


Figure 4: Изменение lab7-1.asm

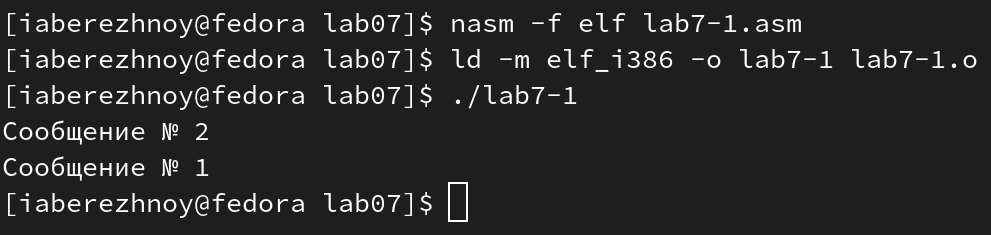


Figure 5: Запуск изменённого файла lab7-1.exe

Снова изменим файл, добавив инструкцию jmp\_label2 после вывода сообщения №3 (рис. [6](#fig:006)). Теперь при запуске программы видим вывод сообщений в обратном порядке (рис. [7](#fig:007)).

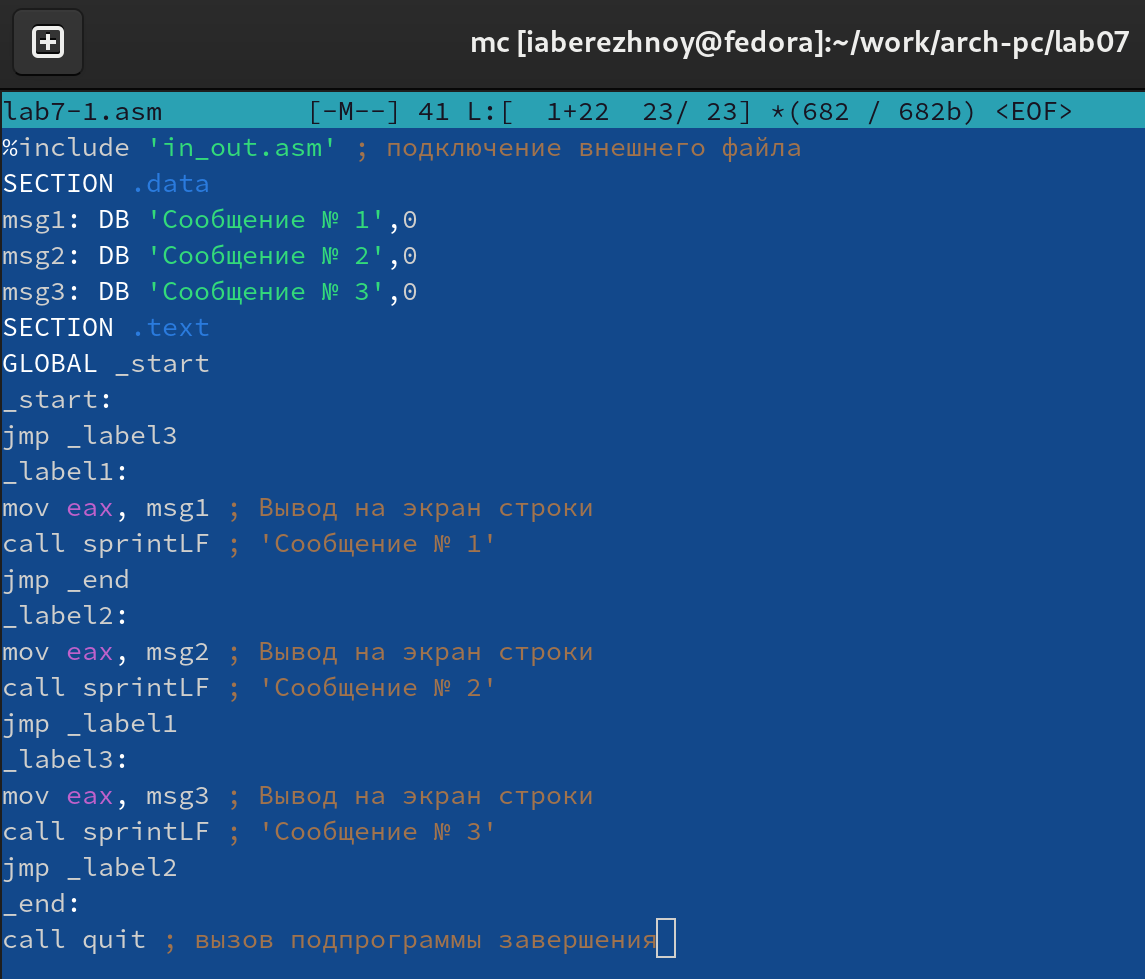


Figure 6: Добавление инструкции jmp\_label2

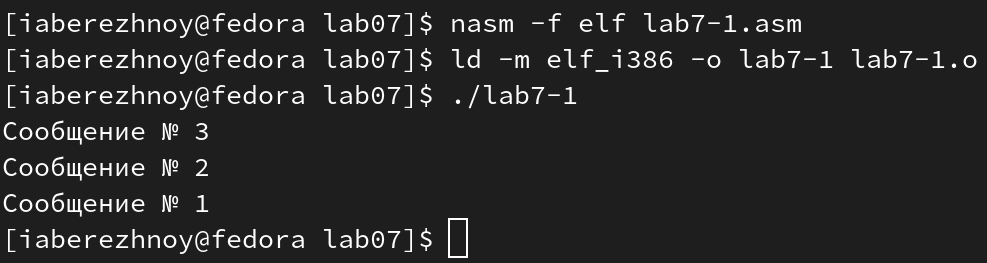


Figure 7: Проверка работы lab7-1.asm

Создадим файл lab7-2.asm и скопируем в него предложенный листинг (рис. [8](#fig:008)). Создадим исполняемый файл и проверим его работу для разных значений B (рис. [9](#fig:009)).

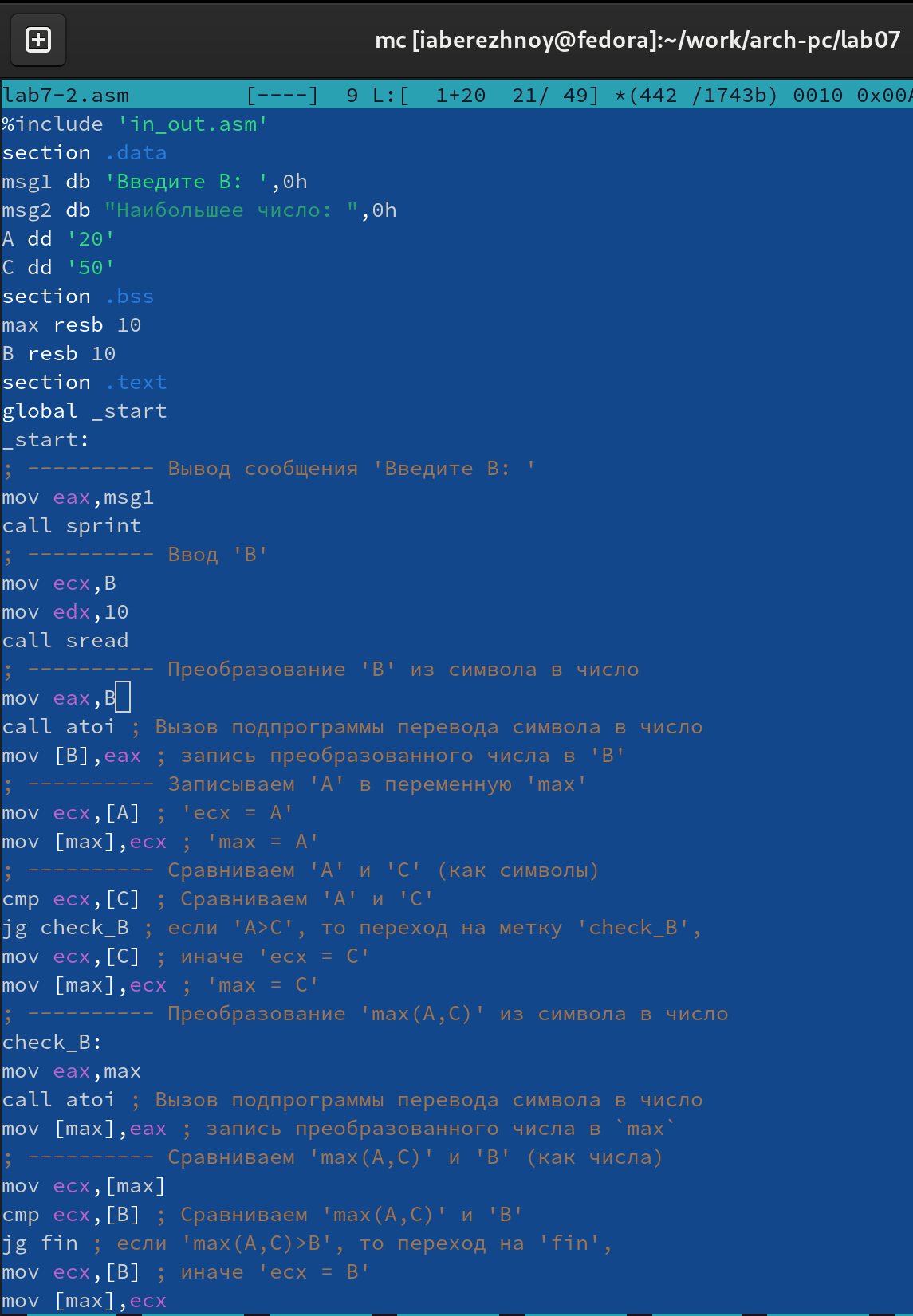


Figure 8: Создание файла lab7-2.asm

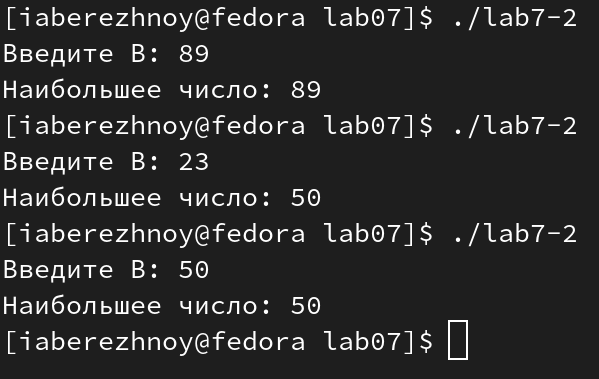


Figure 9: Проверка работы lab7-2.asm

## 3.2 Изучение структуры файла листинга

Создадим файл листинга для программы из файла lab7-2.asm командой nasm -f elf -l lab7-2.lst lab7-2.asm и откроем его (рис. [10](#fig:010)). Разберём несколько строк из этого файла:  
1. Строка 11: jmp nextchar позволяет безусловно перейти к метке nextchar;  
2. Строка 29: mov edx, eax содержит инструкцию mov, а значит значение в регистре eax копируется в регистр edx;  
3. Строка 30: pop eax перемещает последнее значение в стеке в регистр eax.

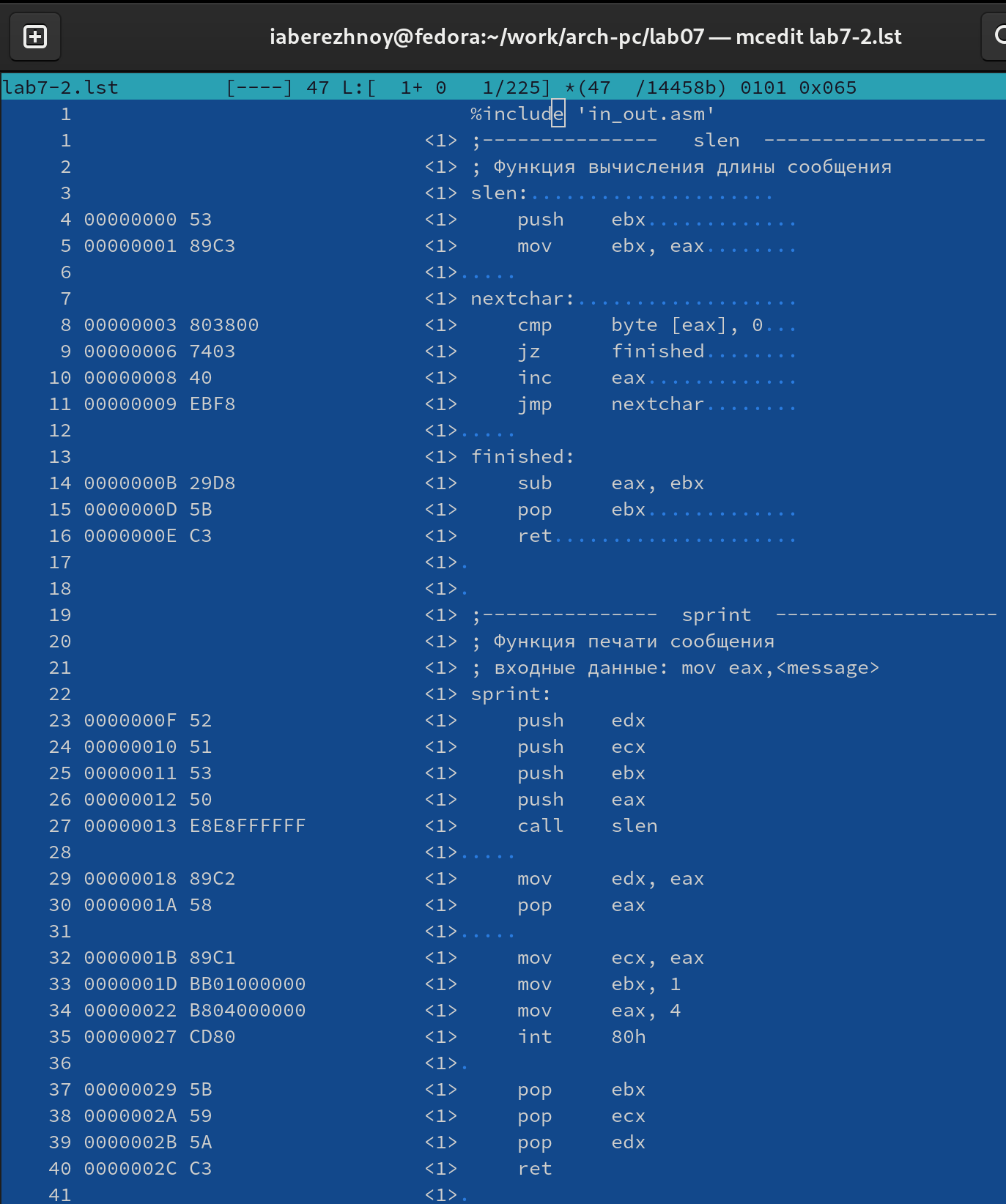


Figure 10: Просмотр файла листинга

Теперь откроем файл с программой lab7-2.asm и в блоке “Сравниваем ‘max(A,C)’ и ‘B’ (как числа)” удалим второй операнд в строке cmp ecx,[B] (рис. [11](#fig:011)). Выполним трансляцию с получением файла листинга и посмотрим, что в нём добавилось (рис. [12](#fig:012)). Видим, что в строке 39 появилась запись ошибки, которая как раз и указывает на то, что мы ввели некорректное число операндов к инструкции.

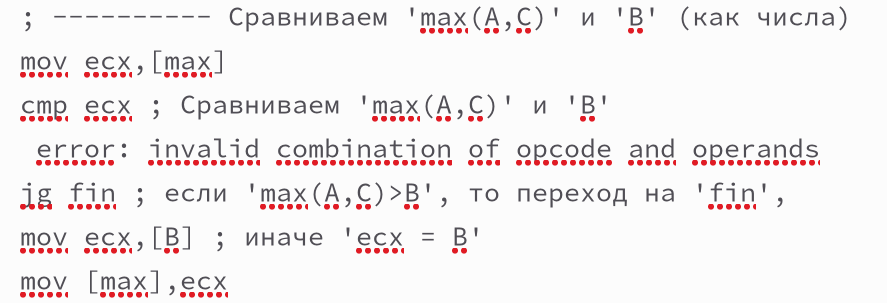


Figure 11: Удаление операнда в lab7-2.asm

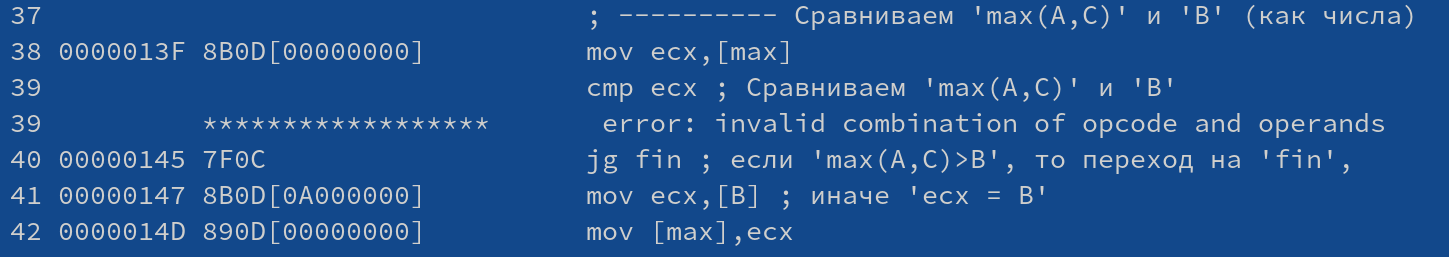


Figure 12: Поиск ошибки в файле листинга

## 3.3 Задание для самостоятельной работы

Создадим файл lab7-3.asm и напишем в нём программу нахождения наименьшей из 3 целочисленных переменных a,b и c в соответствии с вариантом №2, полученным в предыдущей лабораторной работе (рис. [13](#fig:013)). Создадим исполняемый файл и проверим его работу (рис. [14](#fig:014)). Программа работает корректно.

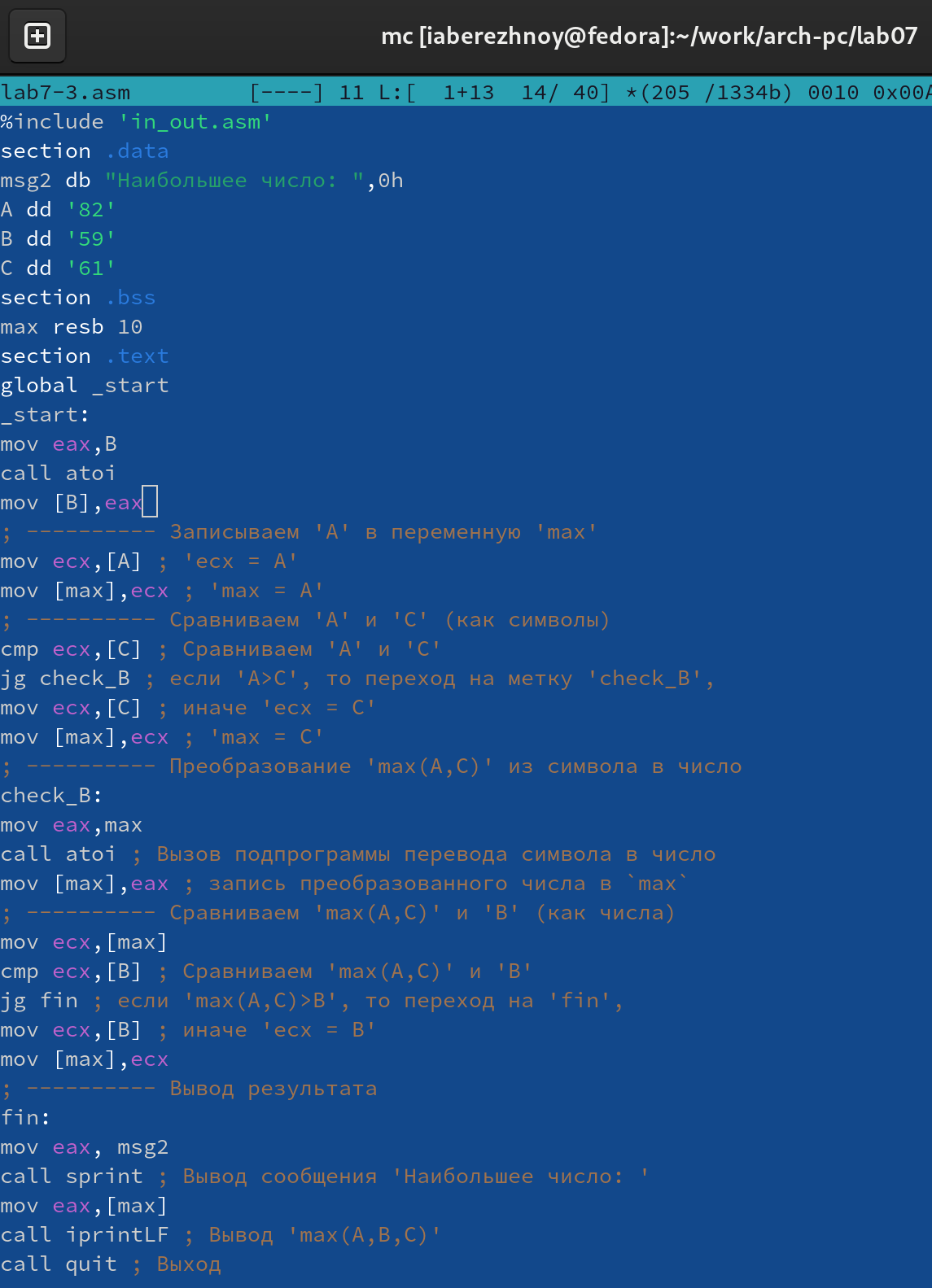


Figure 13: Написание программы №1

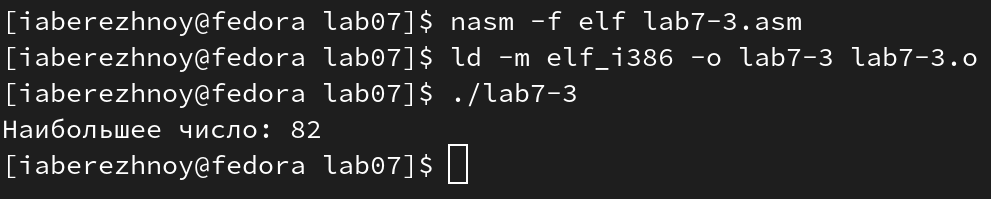


Figure 14: Проверка работы программы №1

**Листинг 7.1. Программа нахождения наибольшей из 3-ёх переменных**

%include 'in\_out.asm'  
section .data  
msg2 db "Наибольшее число: ",0h  
A dd '82'  
B dd '59'  
C dd '61'  
section .bss  
max resb 10  
section .text  
global \_start  
\_start:  
mov eax,B  
call atoi  
mov [B],eax  
; ---------- Записываем 'A' в переменную 'max'  
mov ecx,[A] ; 'ecx = A'  
mov [max],ecx ; 'max = A'  
; ---------- Сравниваем 'A' и 'С' (как символы)  
cmp ecx,[C] ; Сравниваем 'A' и 'С'  
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:

Создадим файл lab7-4.asm и напишем в нём программу, которая для введённых с клавиатуры значений x и a вычисляет значение заданной функции варианта 2 (рис. [15](#fig:015)). Создадим исполняемый файл и проверим его работу (рис. [16](#fig:016)). Программа работает корректно.

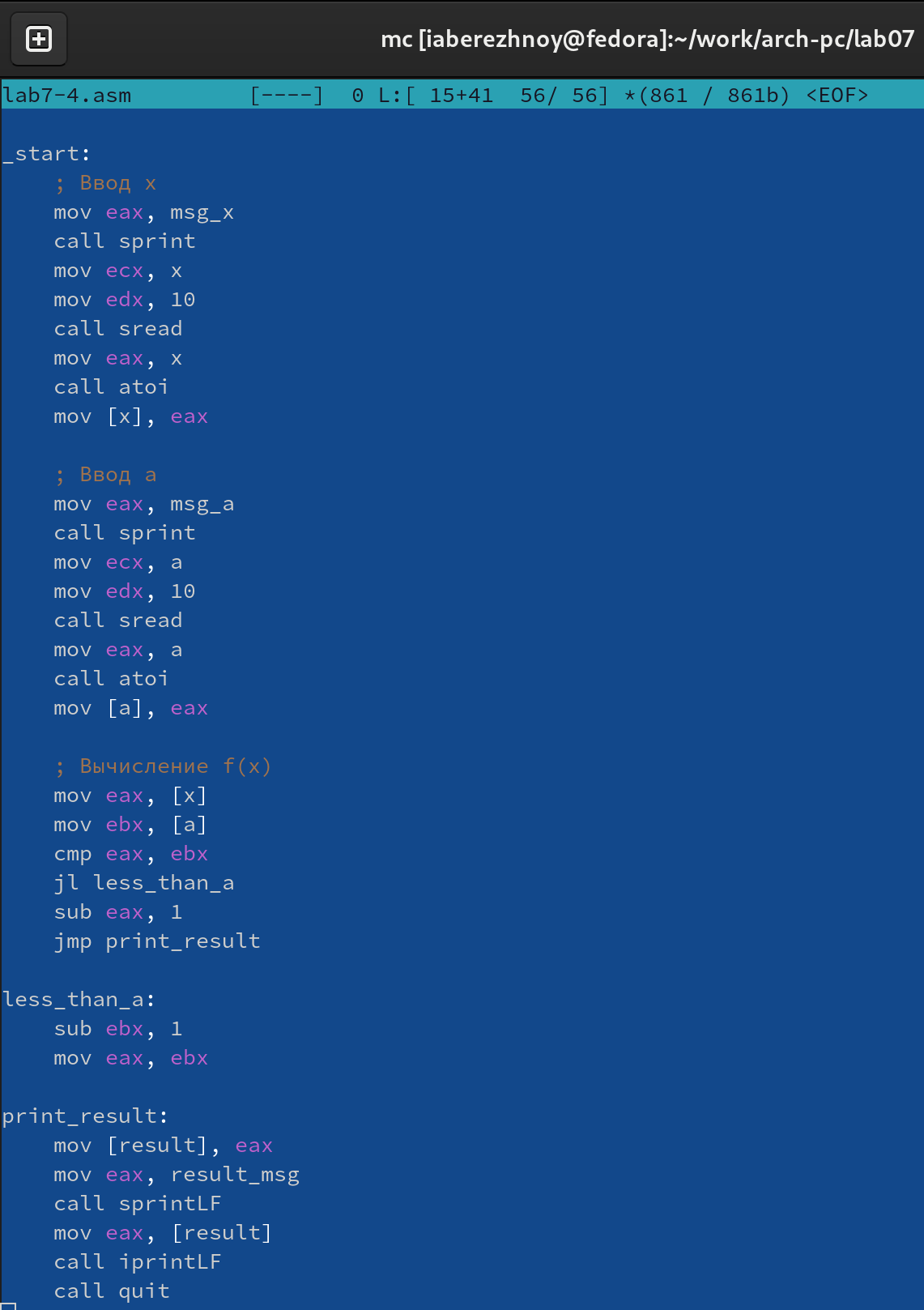


Figure 15: Написание программы №2

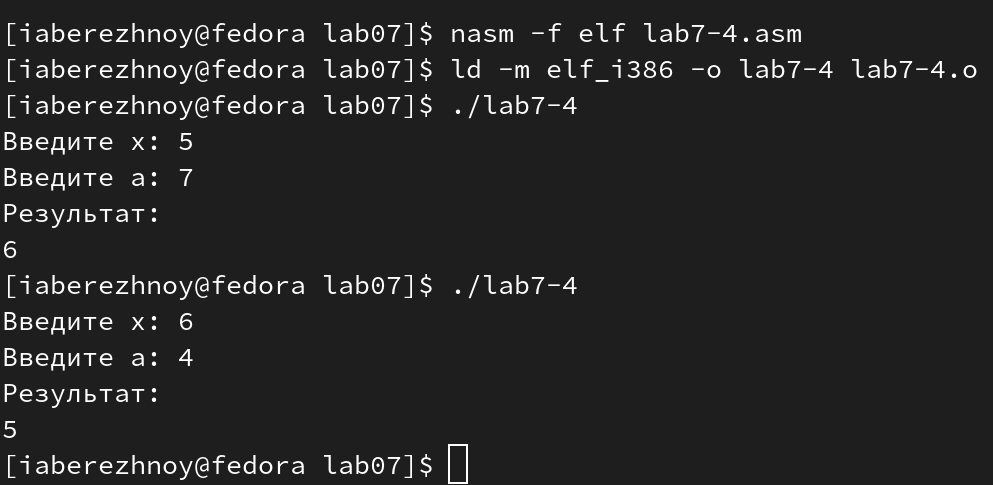


Figure 16: Проверка работы программы №2

**Листинг 7.2. Программа вычисления значения заданной функции**

%include 'in\_out.asm'  
  
section .data  
 msg\_x db 'Введите x: ', 0  
 msg\_a db 'Введите a: ', 0  
 result\_msg db 'Результат: ', 0  
  
section .bss  
 x resb 10  
 a resb 10  
 result resb 10  
  
section .text  
global \_start  
  
\_start:  
 ; Ввод x  
 mov eax, msg\_x  
 call sprint  
 mov ecx, x  
 mov edx, 10  
 call sread  
 mov eax, x  
 call atoi  
 mov [x], eax  
  
 ; Ввод a  
 mov eax, msg\_a  
 call sprint  
 mov ecx, a  
 mov edx, 10  
 call sread  
 mov eax, a  
 call atoi  
 mov [a], eax  
  
 ; Вычисление f(x)  
 mov eax, [x]  
 mov ebx, [a]  
 cmp eax, ebx  
 jl less\_than\_a  
 sub eax, 1  
 jmp print\_result  
  
less\_than\_a:  
 sub ebx, 1  
 mov eax, ebx  
  
print\_result:  
 mov [result], eax  
 mov eax, result\_msg  
 call sprintLF  
 mov eax, [result]  
 call iprintLF  
 call quit

# 4 Выводы

В ходе лабораторной работы мы изучили команды условного и безусловного переходов, приобрели навыки написания программ с использованием переходов, ознакомились с назначением и структурой файла листинга.

# Список литературы

::: [Архитектура ЭВМ](https://esystem.rudn.ru/mod/resource/view.php?id=1030555)