Отчёт по лабораторной работе №7

Команды безусловного и условного перехода в NASM. Программирование ветвлений

Новикова Анастасия Андреевна

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. Реализация переходов в NASM.
2. Изучение структуры файлы листинга.
3. Задания для самостоятельной работы.

# 3 Выполнение лабораторной работы

## 3.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm. (рис. 1).

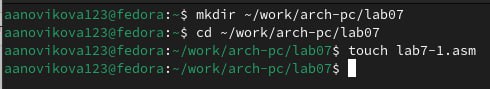


Рис. 1: Создание файлов для лабораторной работы

Ввожу в файл lab7-1.asm текст программы из листинга 7.1. ((рис. 2).



Рис. 2: Ввод текста программы из листинга 7.1

Создаю исполняемый файл и запускаю его. ((рис. 3)

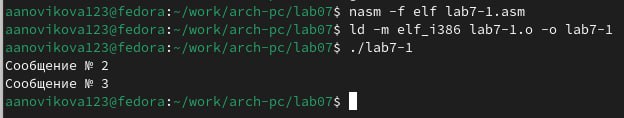


Рис. 3: Запуск программного кода

Таким образом, использование инструкции jmp \_label2 меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки \_label2, пропустив вывод первого сообщения.

Изменю программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого изменяю текст программы в соответствии с листингом 7.2. ((рис. 4)



Рис. 4: Изменение текста программы

Создаю исполняемый файл и проверяю его работу. (рис. 5)

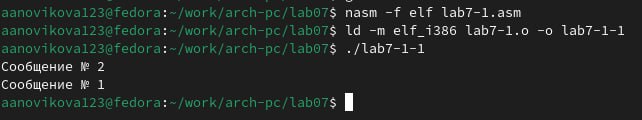


Рис. 5: Создание исполняемого файла

Затем изменяю текст программы, добавив в начале программы jmp \_label3, jmp \_label2 в конце метки jmp \_label3, jmp \_label1 добавляю в конце метки jmp \_label2, и добавляю jmp \_end в конце метки jmp \_label1, (рис. 6)



Рис. 6: Изменение текста программы

чтобы вывод программы был следующим: (рис. 7)

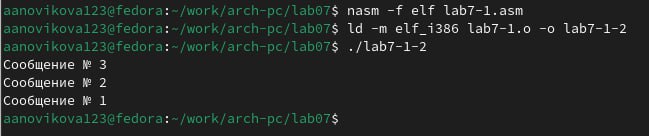


Рис. 7: Вывод программы

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 вводиться с клавиатуры.

Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. (рис. 8)

Создание файла

Рис. 8: Создание файла

Текст программы из листинга 7.3 ввожу в lab7-2.asm. ((рис. 9)

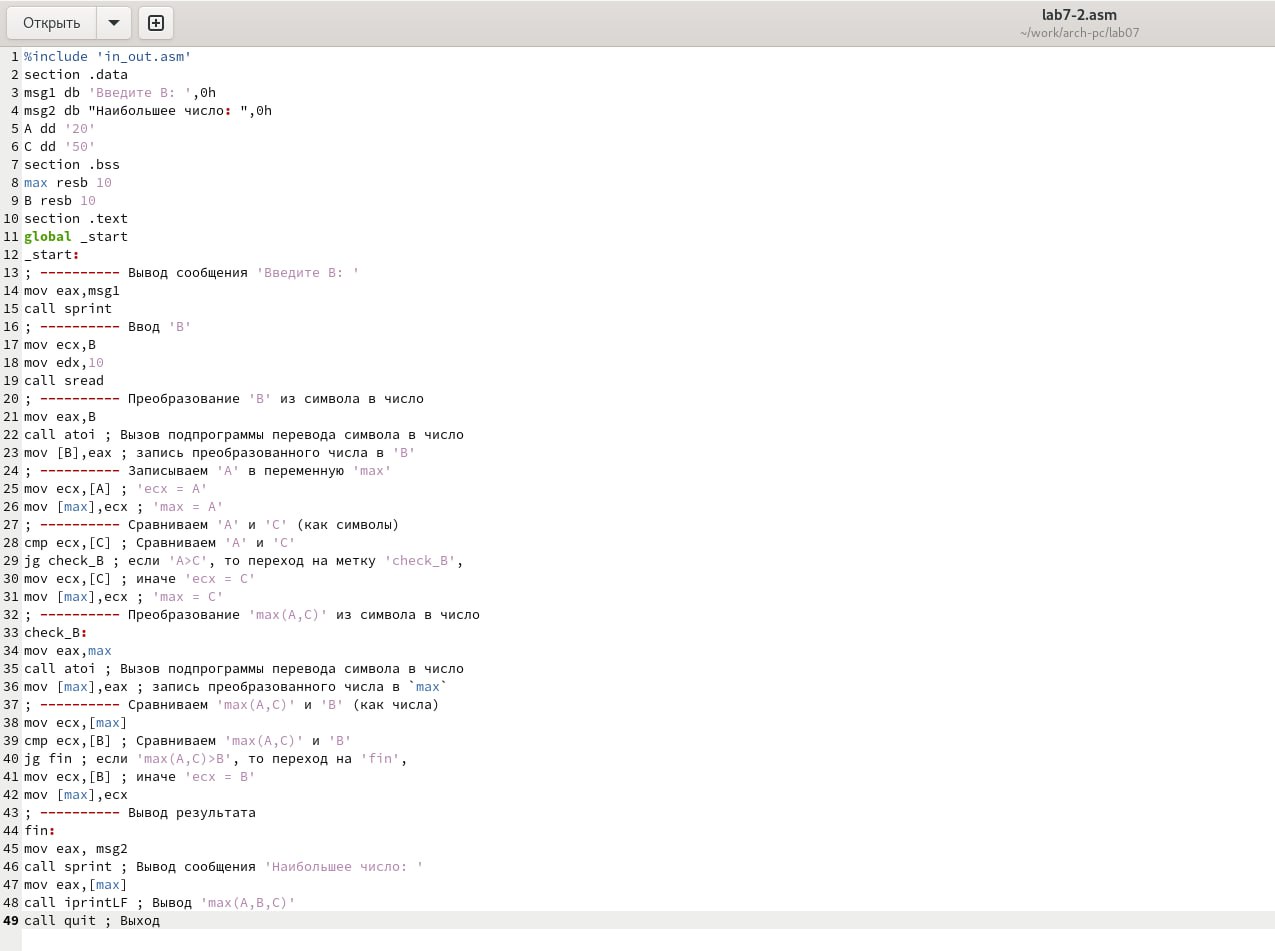


Рис. 9: Ввод текста программы из листинга 7.3

Создаю исполняемый файл и проверьте его работу. (рис. 10)

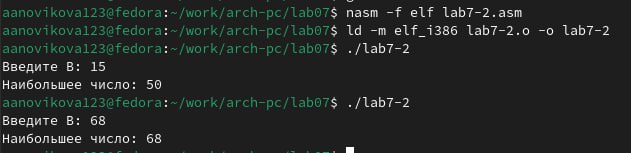


Рис. 10: Проверка работы файла

Файл работает корректно.

## 3.2 Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.asm. (рис. 11)

Создание файла листинга

Рис. 11: Создание файла листинга

Открываю файл листинга lab7-2.lst с помощью текстового редактора и внимательно изучаю его формат и содержимое. (рис. 12)

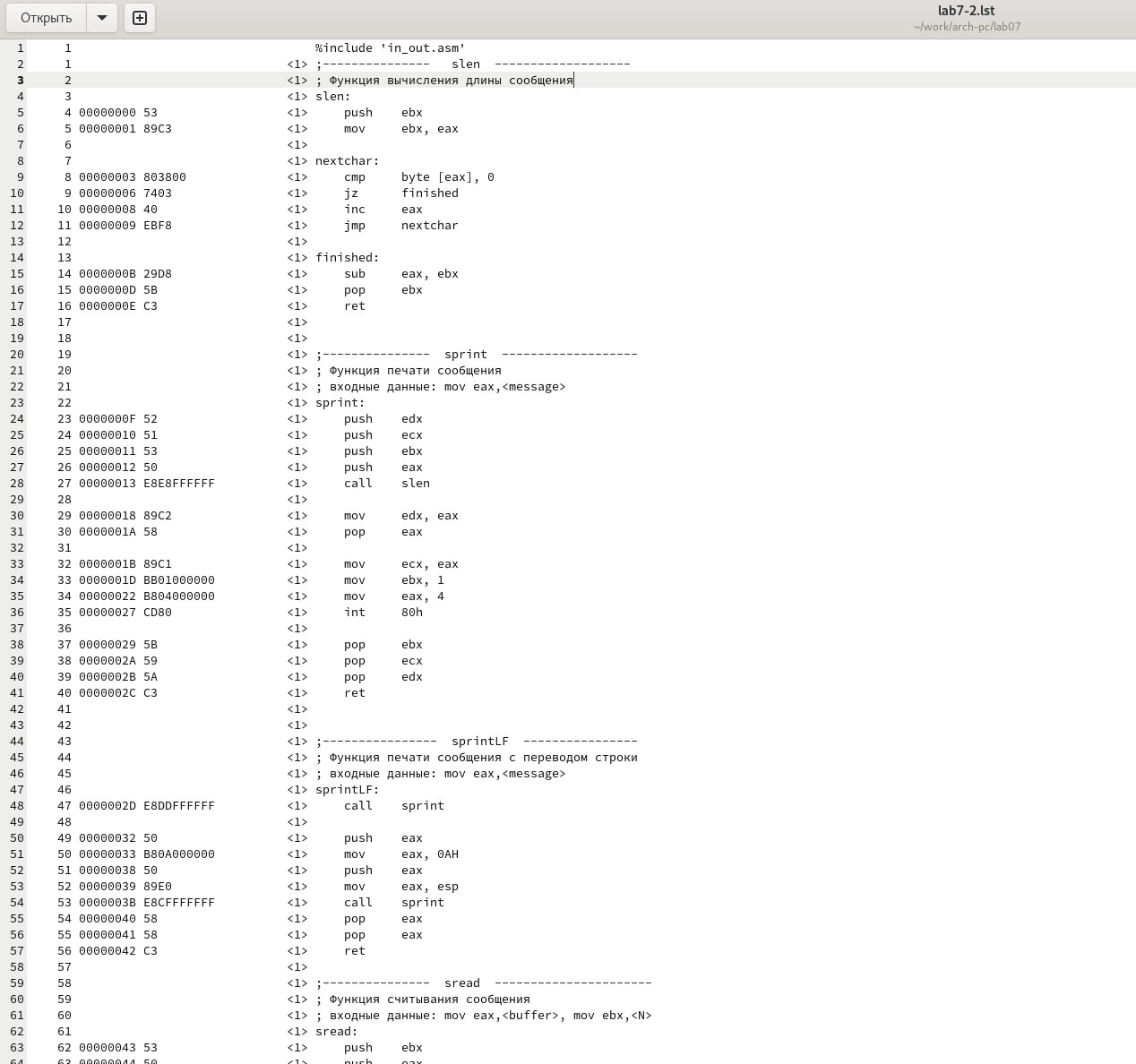


Рис. 12: Изучение файла листинга

В представленных трех строчках содержаться следующие данные: ((рис. 13)

Выбранные строки файла

Рис. 13: Выбранные строки файла

“2” - номер строки кода, “; Функция вычисления длинны сообщения” - комментарий к коду, не имеет адреса и машинного кода.

“3” - номер строки кода, “slen” - название функции, не имеет адреса и машинного кода.

“4” - номер строки кода, “00000000” - адрес строки, “53” - машинный код, “push ebx” - исходный текст программы, инструкция “push” помещает операнд “ebx” в стек.

Открываю файл с программой lab7-2.asm и в выбранной мной инструкции с двумя операндами удаляю выделенный операнд. (рис. 14)

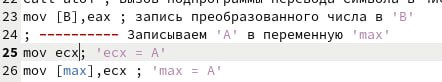


Рис. 14: Удаление выделенного операнда из кода

Выполняю трансляцию с получением файла листинга. (рис. 15)

Получение файла листинга

Рис. 15: Получение файла листинга

Терминал выводит сообщение об ошибке: инструкция mov не может работать, имея только один операнд, из-за чего нарушается работа кода.

Ознакамливаюсь с новым файлом листинга. В файле в этой строчке ошибка. (рис. 16)

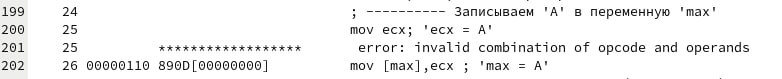


Рис. 16: Ошибка в файле листинга

## 3.3 Задания для самостоятельной работы

*Вариант 20*

*Задание 1*

Пишу программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбираю из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Мой вариант под номером 20, поэтому мои значения - 95, 2, 61. (рис. 17)

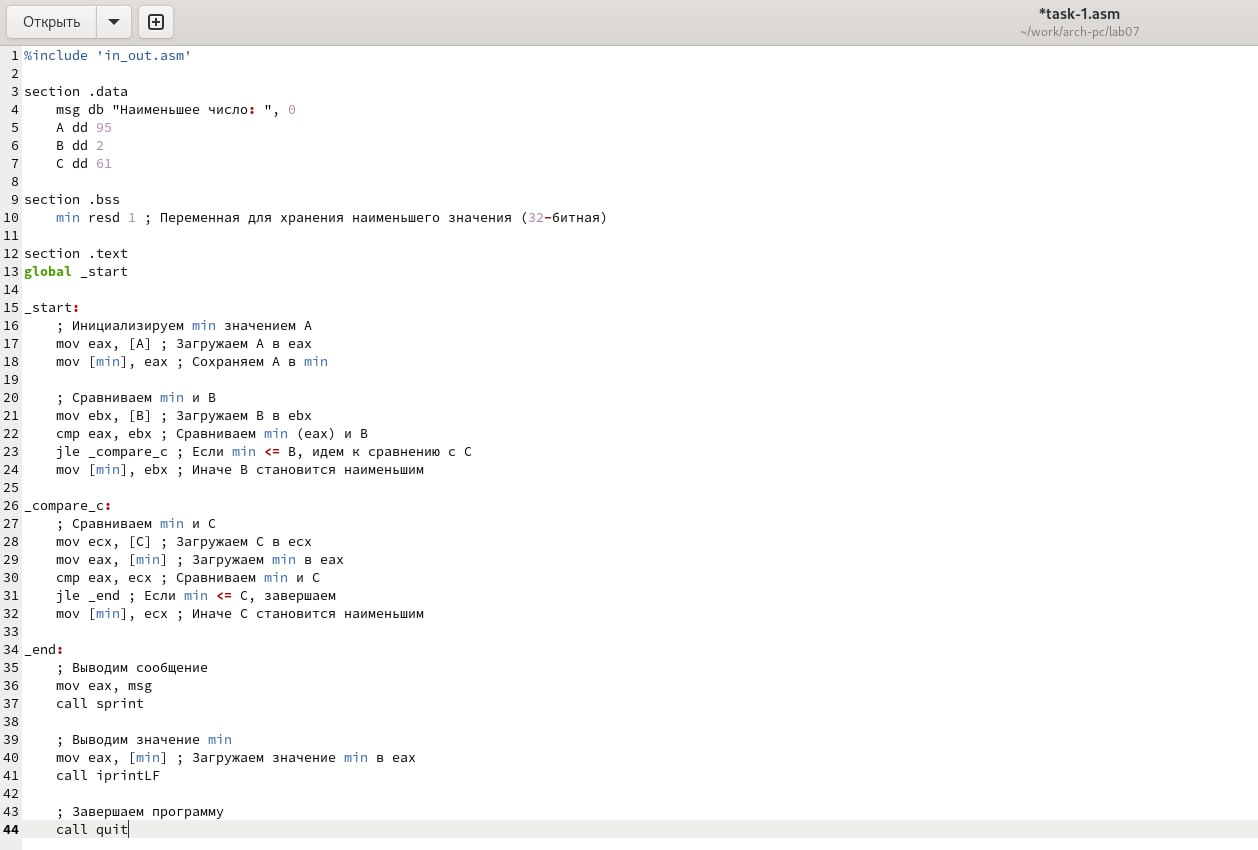


Рис. 17: Написание программы

Создаю исполняемый файл и проверяю его работу, подставляя необходимые значение. (рис. 18)

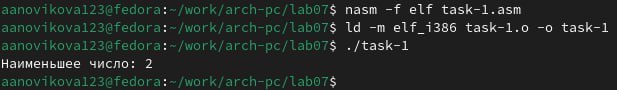


Рис. 18: Запуск файла и проверка его работы

Программа работает корректно.

Код программы:

%include 'in\_out.asm'  
  
section .data  
 msg db "Наименьшее число: ", 0  
 A dd 95   
 B dd 2   
 C dd 61   
  
section .bss  
 min resd 1 ; Переменная для хранения наименьшего значения (32-битная)  
  
section .text  
global \_start  
  
\_start:  
 ; Инициализируем min значением A  
 mov eax, [A] ; Загружаем A в eax  
 mov [min], eax ; Сохраняем A в min  
  
 ; Сравниваем min и B  
 mov ebx, [B] ; Загружаем B в ebx  
 cmp eax, ebx ; Сравниваем min (eax) и B  
 jle \_compare\_c ; Если min <= B, идем к сравнению с C  
 mov [min], ebx ; Иначе B становится наименьшим  
  
\_compare\_c:  
 ; Сравниваем min и C  
 mov ecx, [C] ; Загружаем C в ecx  
 mov eax, [min] ; Загружаем min в eax  
 cmp eax, ecx ; Сравниваем min и C  
 jle \_end ; Если min <= C, завершаем  
 mov [min], ecx ; Иначе C становится наименьшим  
  
\_end:  
 ; Выводим сообщение  
 mov eax, msg  
 call sprint  
  
 ; Выводим значение min  
 mov eax, [min] ; Загружаем значение min в eax  
 call iprintLF  
  
 ; Завершаем программу  
 call quit

*Задание 2*

Пишу программу (рис. 19), которая для введенных с клавиатуры значений х и а вычисляет значение и выводит результат вычислений заданной для моего варианта функции f(x): *x - a, если x >= a;*

*5, если x < a*.

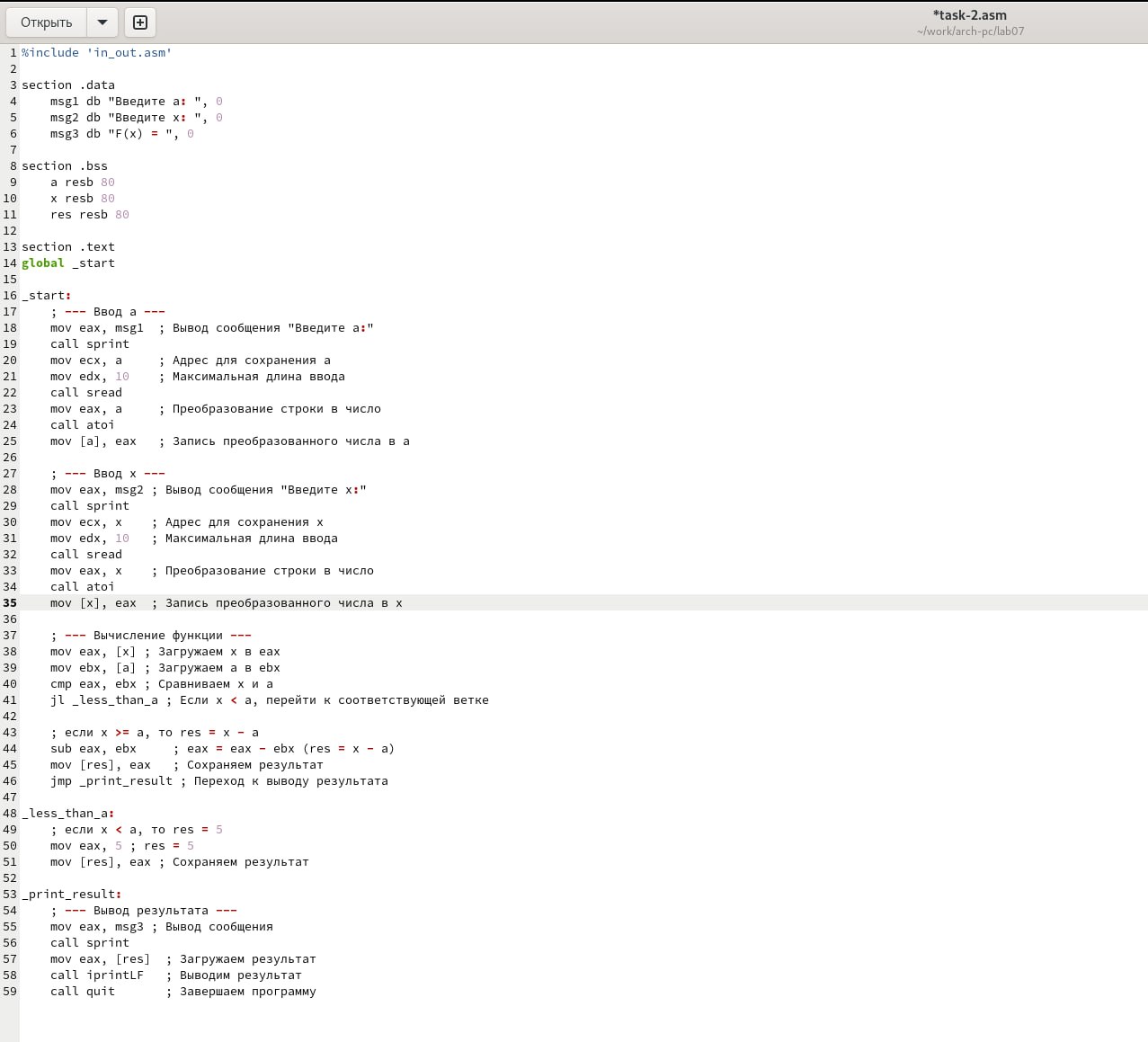


Рис. 19: Написание программы

Создаю исполняемый файл и проверяю его работу для значений х и а соответственно: (1, 2), (2, 1). (рис. 20)

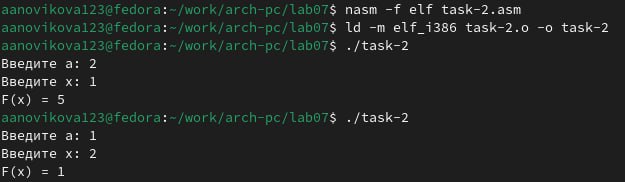


Рис. 20: Запуск файла и проверка его работы

Программа работает корректно.

Код программы:

%include 'in\_out.asm'  
  
section .data  
 msg1 db "Введите a: ", 0  
 msg2 db "Введите x: ", 0  
 msg3 db "F(x) = ", 0  
  
section .bss  
 a resb 80   
 x resb 80  
 res resb 80  
  
section .text  
global \_start  
  
\_start:  
 ; --- Ввод a ---  
 mov eax, msg1 ; Вывод сообщения "Введите a:"  
 call sprint  
 mov ecx, a ; Адрес для сохранения a  
 mov edx, 10 ; Максимальная длина ввода  
 call sread  
 mov eax, a ; Преобразование строки в число  
 call atoi  
 mov [a], eax ; Запись преобразованного числа в а  
  
 ; --- Ввод x ---  
 mov eax, msg2 ; Вывод сообщения "Введите x:"  
 call sprint  
 mov ecx, x ; Адрес для сохранения x  
 mov edx, 10 ; Максимальная длина ввода  
 call sread  
 mov eax, x ; Преобразование строки в число  
 call atoi  
 mov [x], eax ; Запись преобразованного числа в х  
  
 ; --- Вычисление функции ---  
 mov eax, [x] ; Загружаем x в eax  
 mov ebx, [a] ; Загружаем a в ebx  
 cmp eax, ebx ; Сравниваем x и a  
 jl \_less\_than\_a ; Если x < a, перейти к соответствующей ветке  
  
 ; если x >= a, то res = x - a  
 sub eax, ebx ; eax = eax - ebx (res = x - a)  
 mov [res], eax ; Сохраняем результат  
 jmp \_print\_result ; Переход к выводу результата  
  
\_less\_than\_a:  
 ; если x < a, то res = 5  
 mov eax, 5 ; res = 5  
 mov [res], eax ; Сохраняем результат  
  
\_print\_result:  
 ; --- Вывод результата ---  
 mov eax, msg3 ; Вывод сообщения  
 call sprint  
 mov eax, [res] ; Загружаем результат  
 call iprintLF ; Выводим результат  
 call quit ; Завершаем программу

# 4 Выводы

По итогам данной лабораторной работы были изучены команды условного и безусловного переходов, приобретены навыки написания программ с использованием переходов, а также было знакомство с назначением и структурой файла листинга.