Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютера

Барбакова Алиса Саяновна

Содержание

# 1 Цель работы

Цель данной лабораторной работы - изучение команд условного и безусловного переходов, приобретение навыков написания программ с использованием переходов, знакомство с назначением и структурой файла листинга.

# 2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

# 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 про- граммы без каких-либо условий.  
Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, созда- ваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.  
Структура листинга: • номер строки — это номер строки файла листинга (нужно помнить, что номер строки в файле листинга может не соответствовать номеру строки в файле с исходным текстом программы);  
• адрес — это смещение машинного кода от начала текущего сегмента;  
• машинный код представляет собой ассемблированную исходную строку в виде шестна- дцатеричной последовательности. (например, инструкция int 80h начинается по сме- щению 00000020 в сегменте кода; далее идёт машинный код, в который ассемблируется инструкция, то есть инструкция int 80h ассемблируется в CD80 (в шестнадцатеричном представлении); CD80 — это инструкция на машинном языке, вызывающая прерывание ядра);  
• исходный текст программы — это просто строка исходной программы вместе с ком- ментариями (некоторые строки на языке ассемблера, например, строки, содержащие только комментарии, не генерируют никакого машинного кода, и поля «смещение» и «исходный текст программы» в таких строках отсутствуют, однако номер строки им присваивается).

# 4 Выполнение лабораторной работы

## 4.1 Реализация переходов в NASM

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

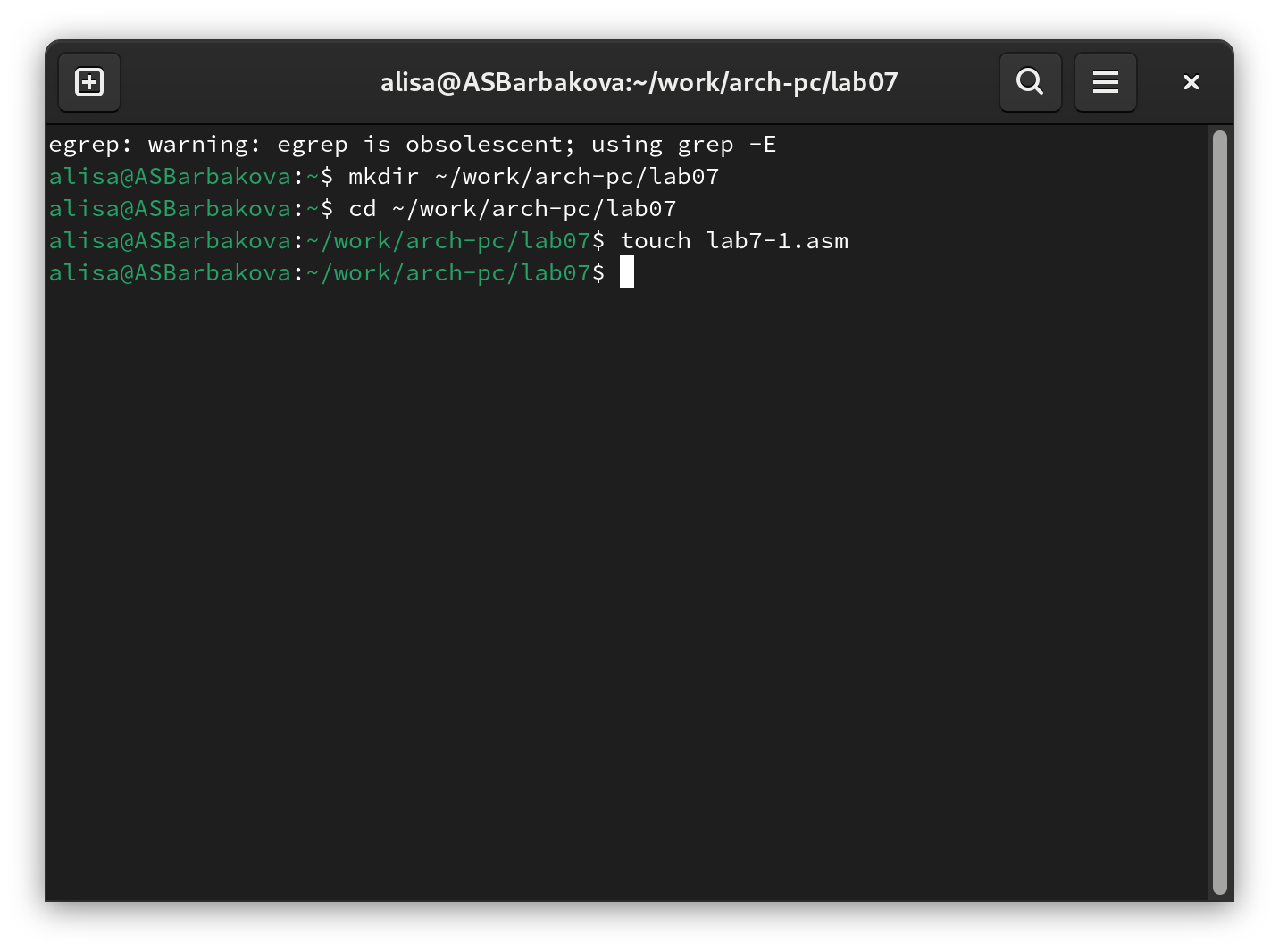


Рис. 1: Создание каталога и файла для программы

Копирую в текущий каталог файл in\_out.asm (рис. 2).

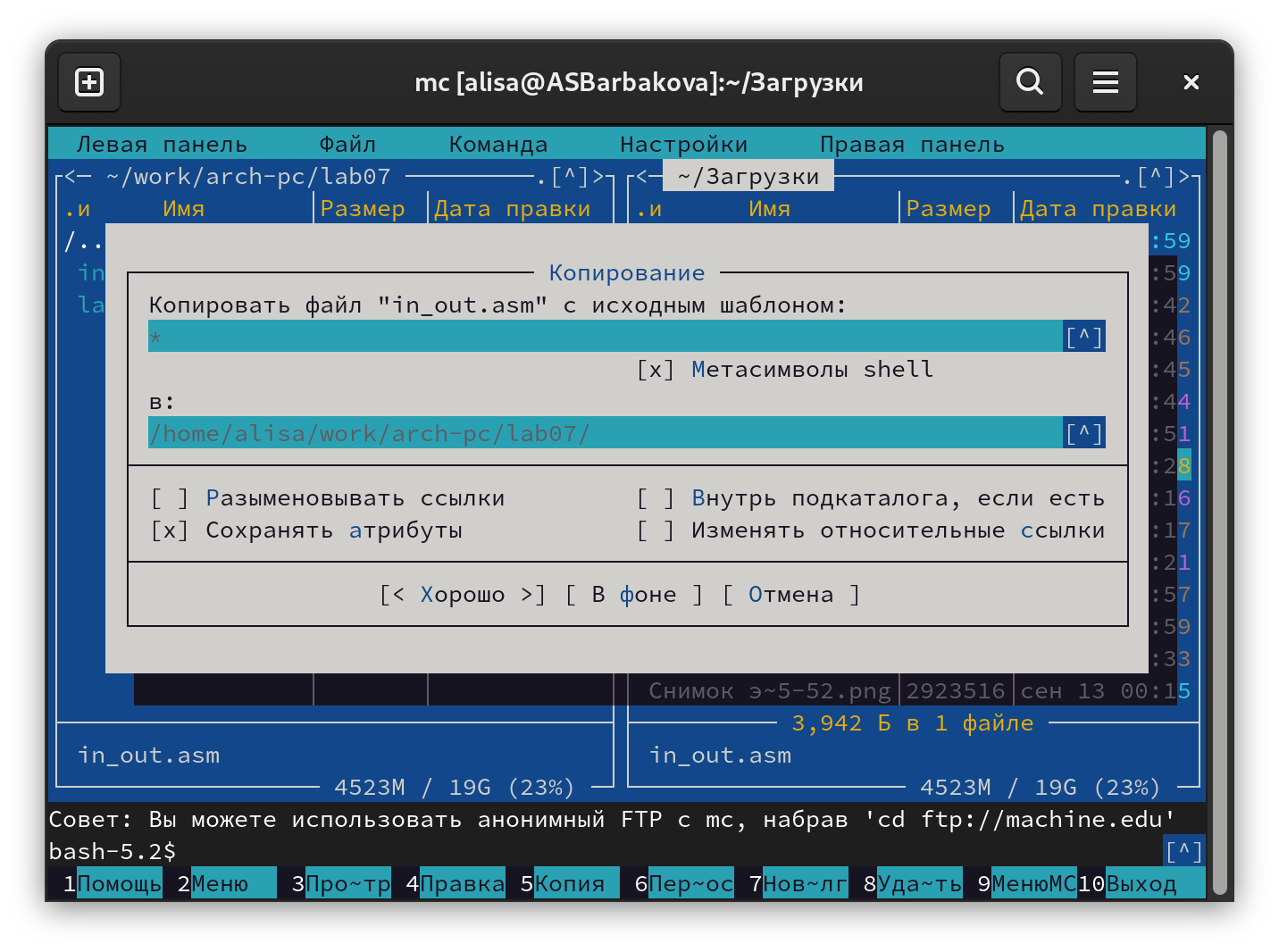


Рис. 2: Создание копии файла

Копирую код из листинга в файл будущей программы. (рис. -fig. 3).

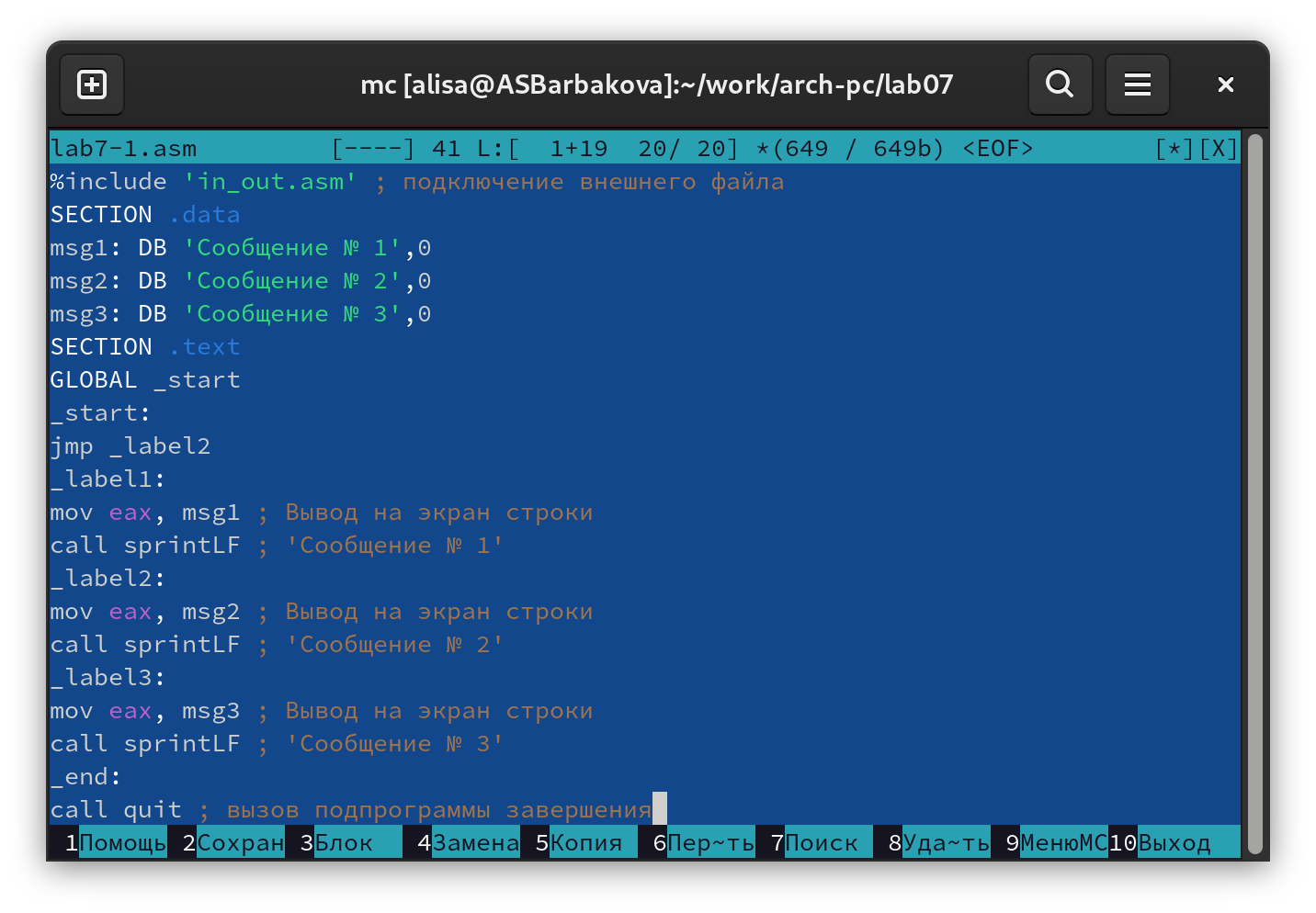


Рис. 3: Сохранение программы

Запускаю программу. Вижу, что неусловный переход действительно изменяет порядок выполнения инструкций (рис. -fig. 4).

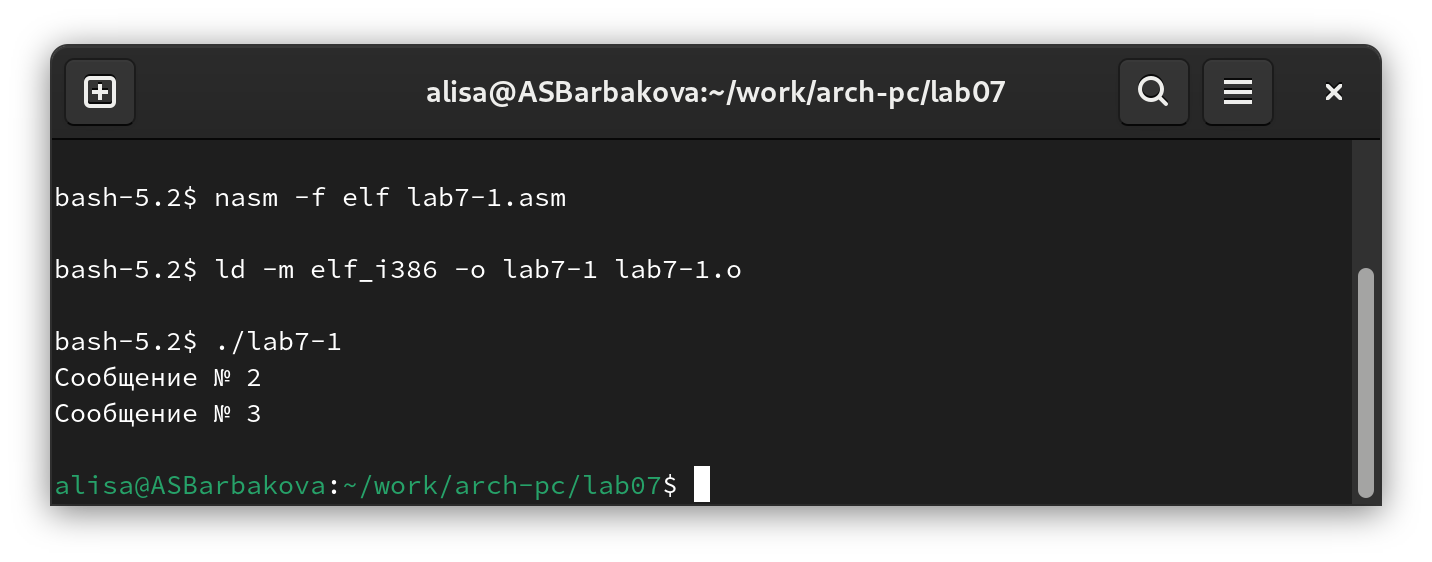


Рис. 4: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения функций (рис. -fig. 5).

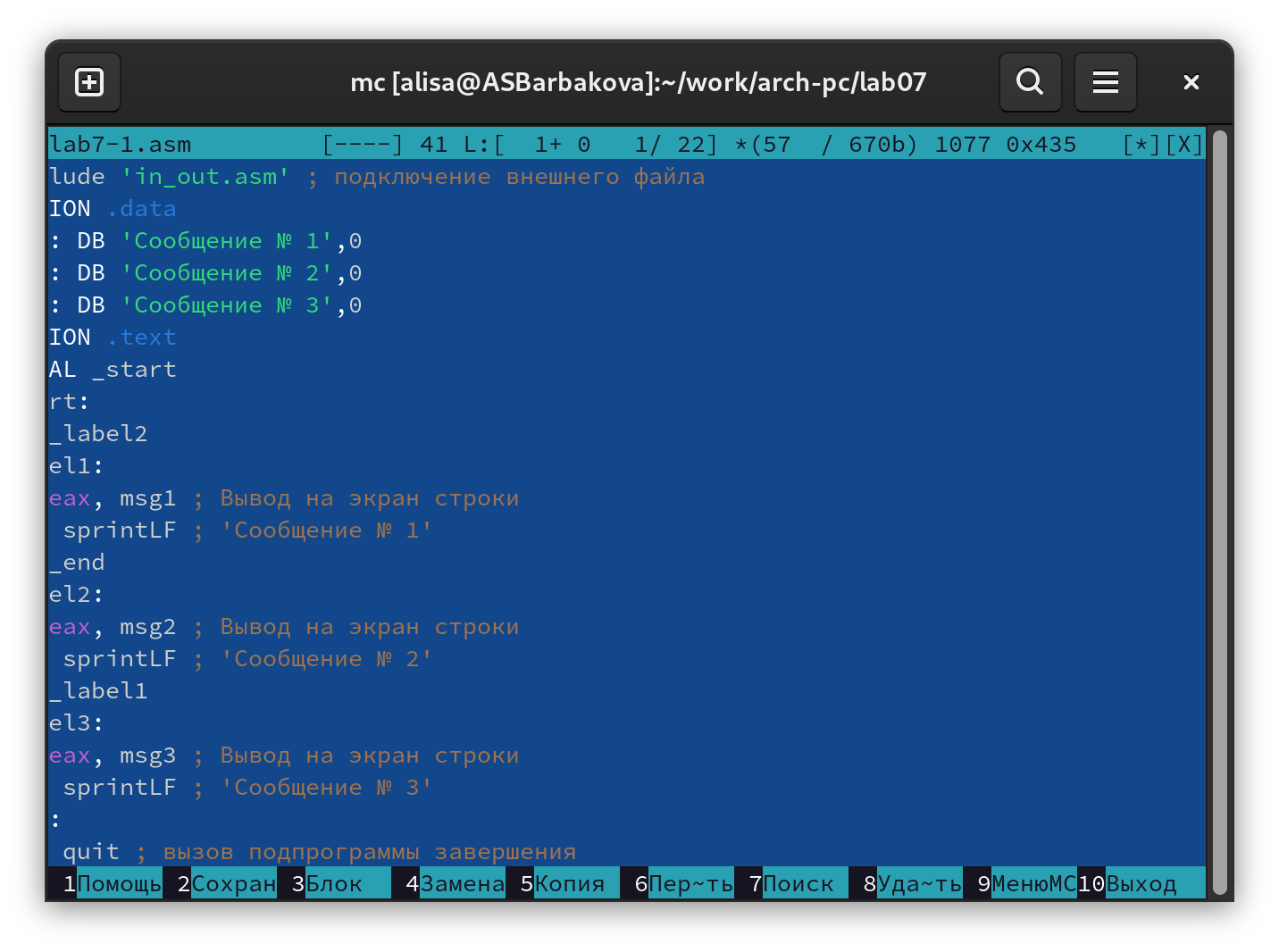


Рис. 5: Изменение программы

Запускаю программу и проверяю, что примененные изменения верны (рис. -fig. 6).

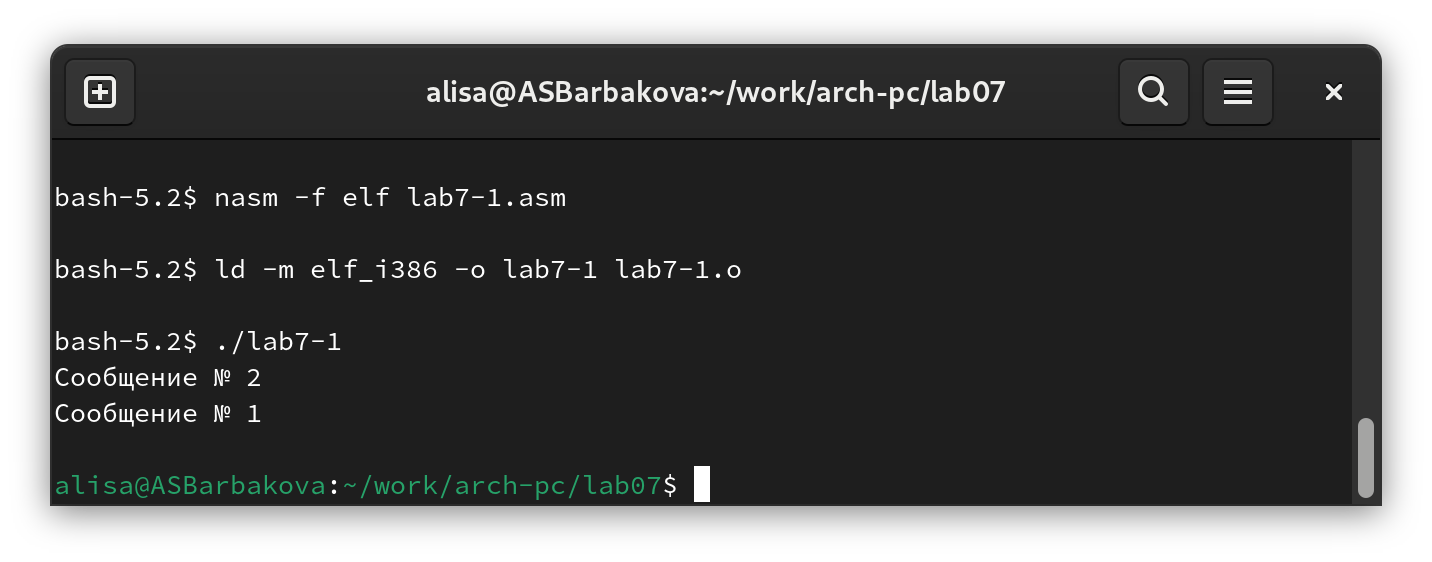


Рис. 6: Запуск измененной программы

Далее я изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. -fig. 7).

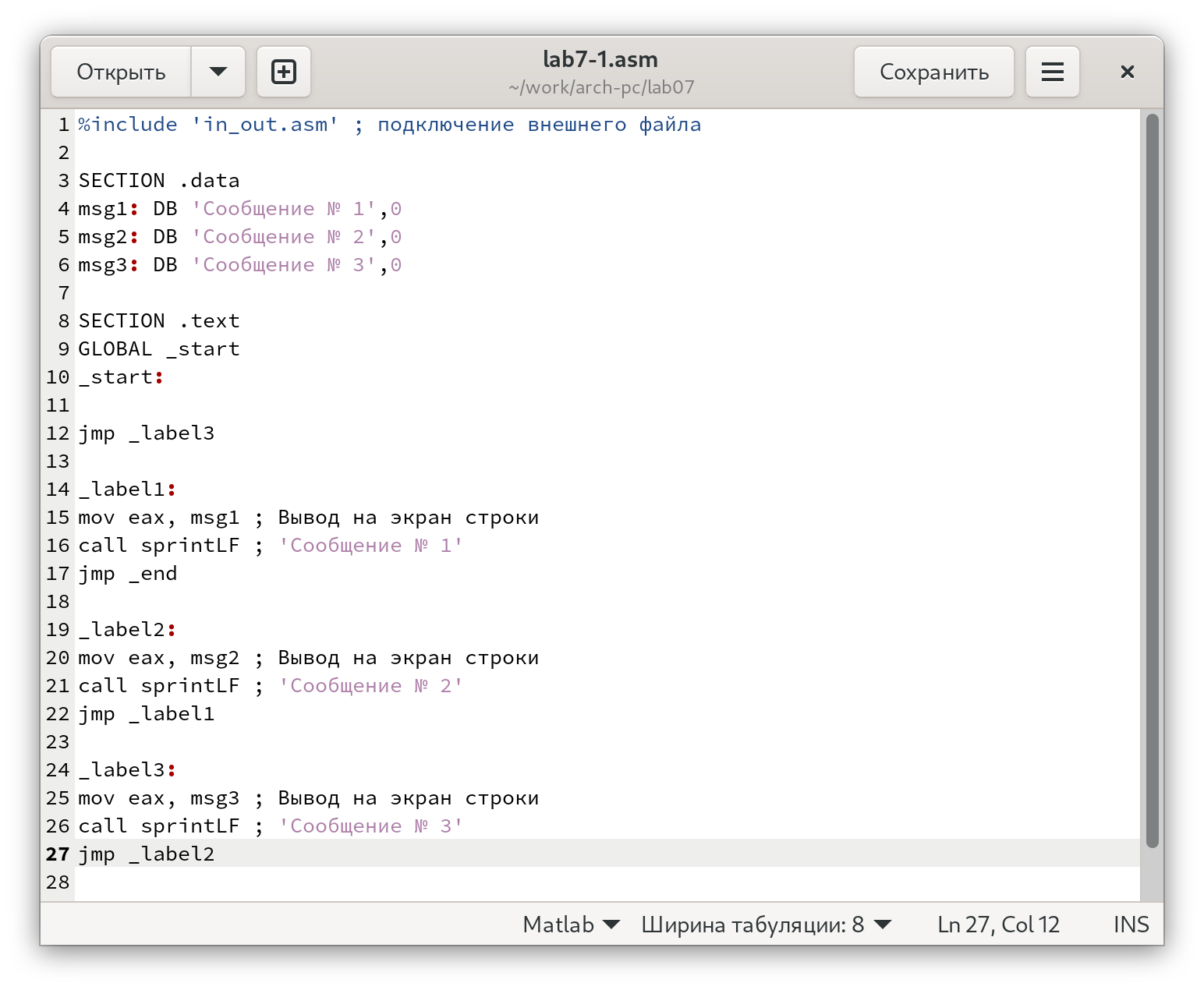


Рис. 7: Изменение программы

Запускаю программу, сообщения выводятся в нужном мне порядке (рис. -fig. 8).

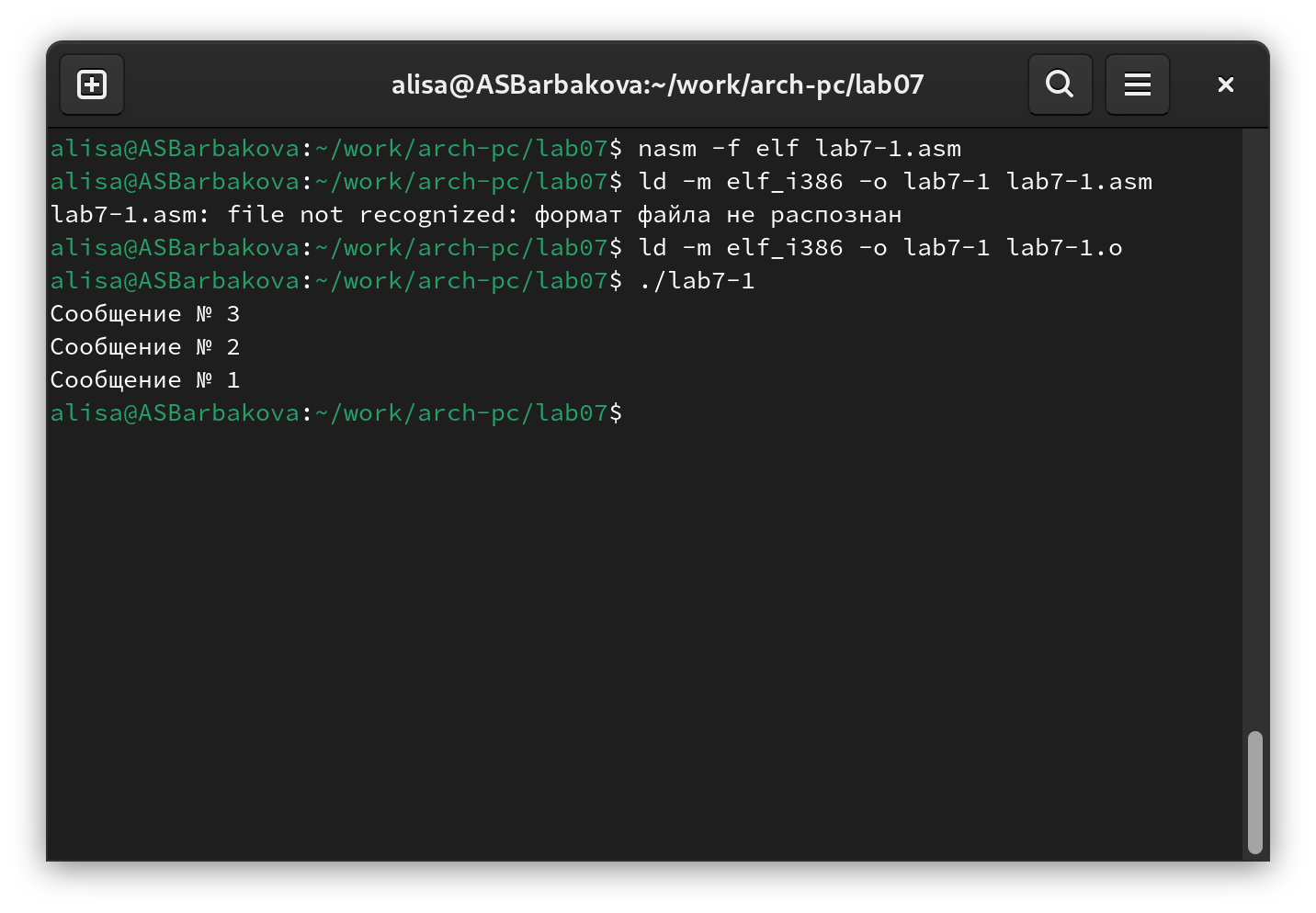


Рис. 8: Проверка изменений

Создаю новый рабочий файл lab7-2.asm и вставляю в него код из следующего листинга (рис. -fig. 9).

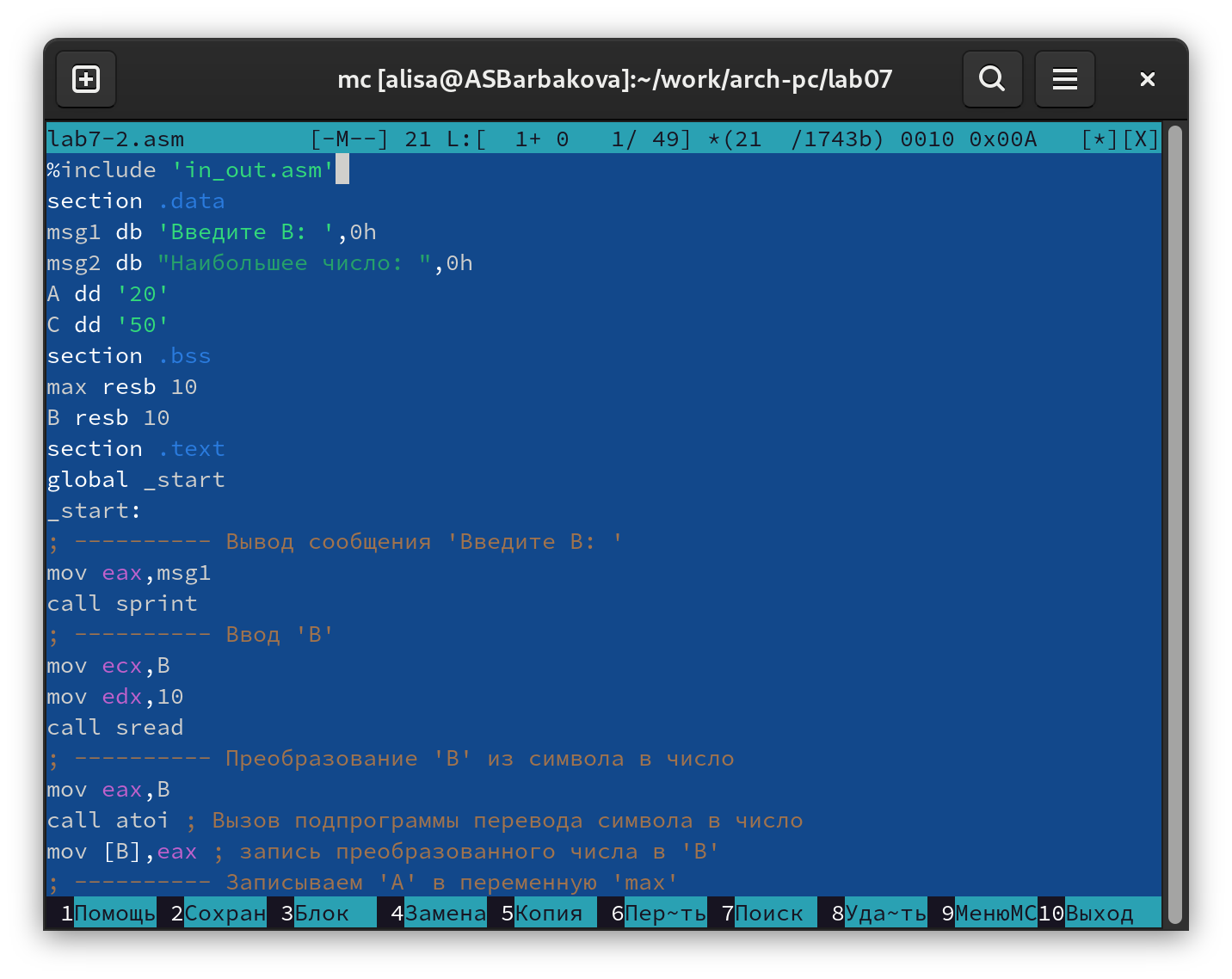


Рис. 9: Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными (рис. -fig. 10).

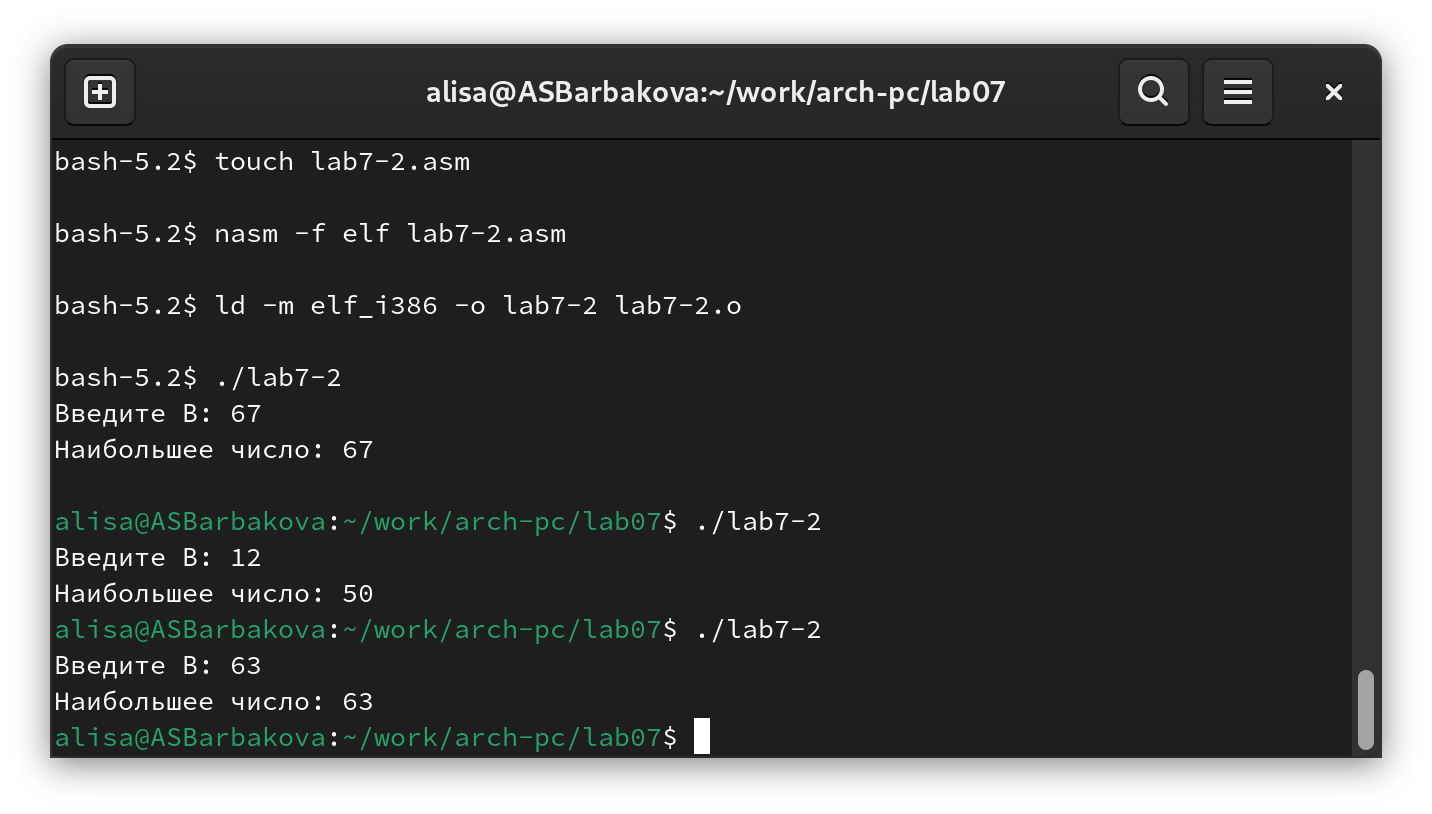


Рис. 10: Проверка программы из листинга

## 4.2 Изучение структуры файла листинга

Создаю файл листинга с помощью команды nasm -f elf -l lab7-2.lst lab7-2.asm и открываю его с помощью текстового редактора mcedit (рис. -fig. 11).

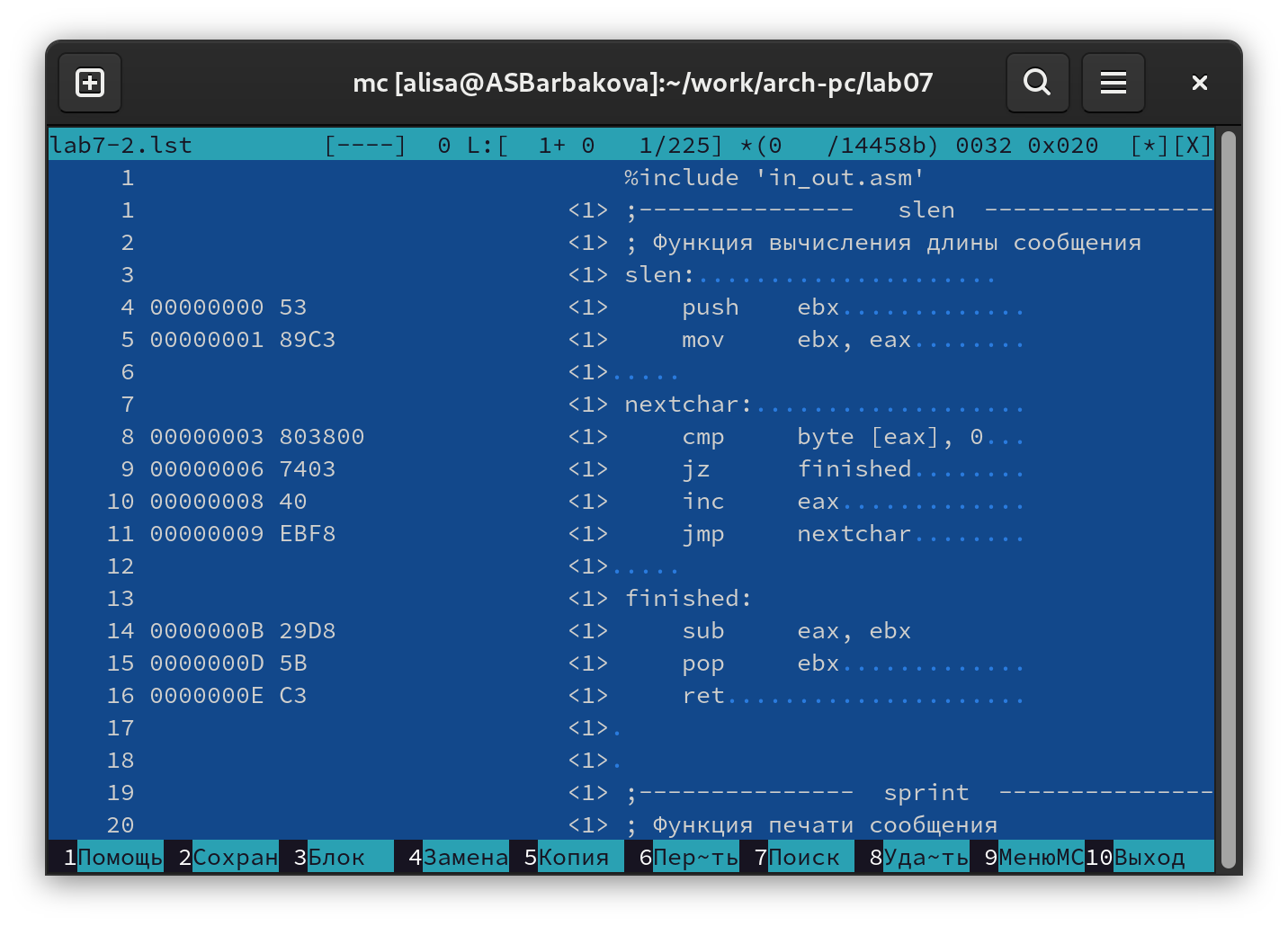


Рис. 11: Проверка файла листинга

1. Строка 4: 00000000 53 <1> push ebx…………  
   Адрес: 00000000 — это адрес в памяти, где находится инструкция. Код операции: 53 — это байт-код ассемблерной инструкции push ebx. Эта инструкция помещает значение регистра ebx в стек.
2. Строка 5: 00000001 89C3 <1> mov ebx, eax……..  
   Адрес: 00000001 — адрес в памяти, где эта инструкция будет выполнена. Код операции: 89C3 — это байт-код для инструкции mov ebx, eax. Инструкция копирует содержимое регистра eax в регистр ebx.
3. Строка 8: 00000003 803800 <1> cmp byte [eax], 0…  
   Адрес: 00000003 — адрес этой инструкции в памяти. Код операции: 803800 — это собственный код для инструкции cmp byte [eax], 0. Эта инструкция сравнивает байт, на который указывает eax, с нулем.  
   Эти три строки из листинга вычисляет длину строки. Первая строка сохраняет состояние регистра, вторая сохраняет начало строки, а третья проверяет каждый символ строки на равенство нулю, что позволяет завершить вычисление длины строки.

Удаляю один операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. -fig. 12).

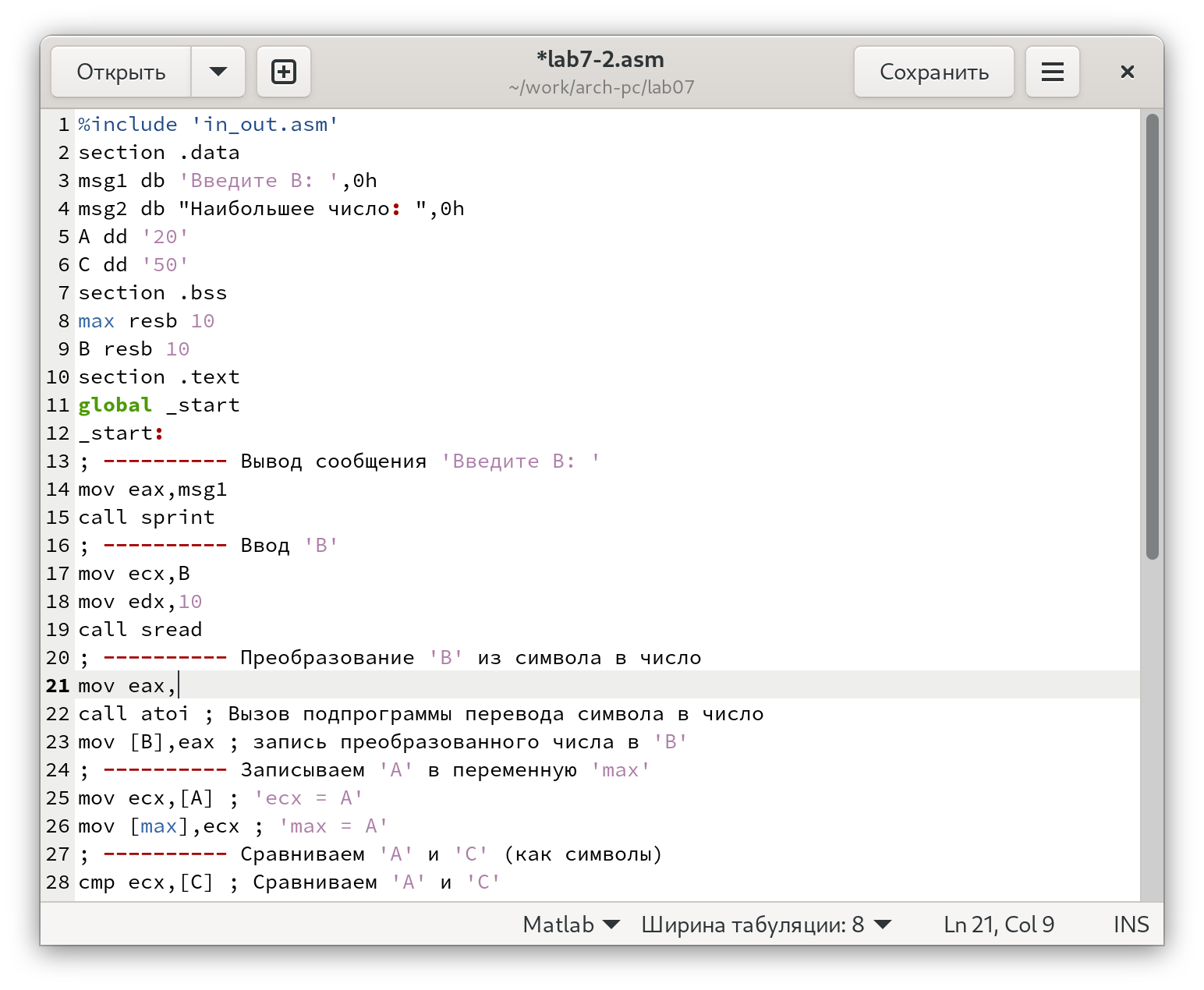


Рис. 12: Удаление операнда из программы

Новый файл листинга у меня не создаётся, выходит ошибка при попытке трансляции файла. Никакие выходные файлы не создаются. (рис. -fig. 13).

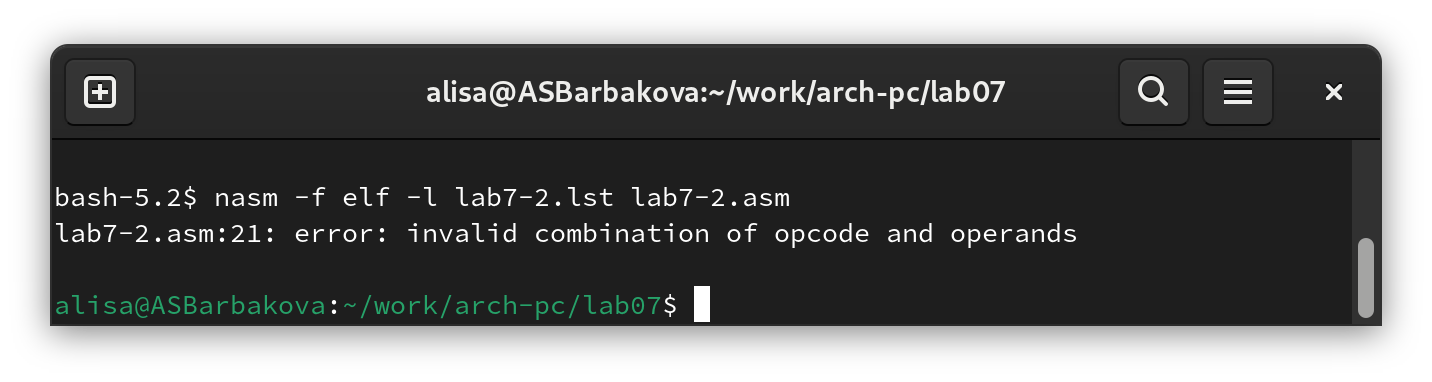


Рис. 13: Просмотр ошибки

## 4.3 Задания для самостоятельной работы

Для выполнения самостоятельной работы использую свой вариант - восьмой - из лабораторной работы №6. Копирую файл lab7-2.asm и изменяю программу так, чтобы она выводила переменную с наименьшим значением (рис. -fig. 14).

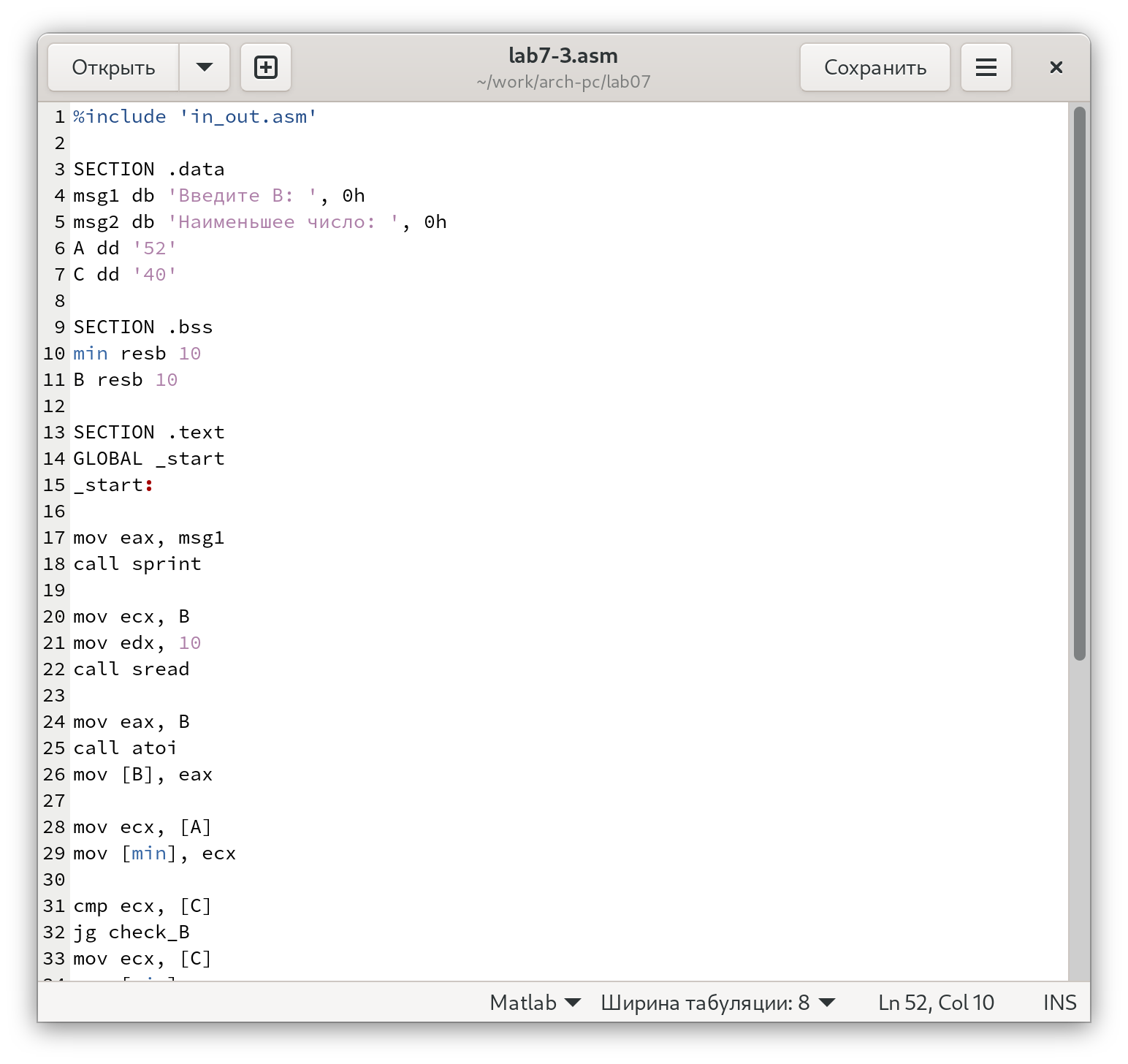


Рис. 14: Первая программа самостоятельной работы

Код первой программы:

%include 'in\_out.asm'  
  
SECTION .data  
msg1 db 'Введите В: ', 0h  
msg2 db 'Наименьшее число: ', 0h  
A dd '52'  
C dd '40'  
  
SECTION .bss  
min resb 10  
B resb 10  
  
SECTION .text  
GLOBAL \_start  
\_start:  
  
mov eax, msg1  
call sprint  
  
mov ecx, B  
mov edx, 10  
call sread  
  
mov eax, B  
call atoi  
mov [B], eax  
  
mov ecx, [A]  
mov [min], ecx  
  
cmp ecx, [C]  
jg check\_B  
mov ecx, [C]  
mov [min], ecx  
  
check\_B:  
mov eax, min  
call atoi  
mov [min], eax  
  
mov ecx, [min]  
cmp ecx, [B]  
jb fin  
mov ecx, [B]  
mov [min], ecx  
  
fin:  
mov eax, msg2  
call sprint  
mov eax, [min]  
call iprintLF  
call quit

Проверяю корректность написания первой программы (рис. -fig. 15).

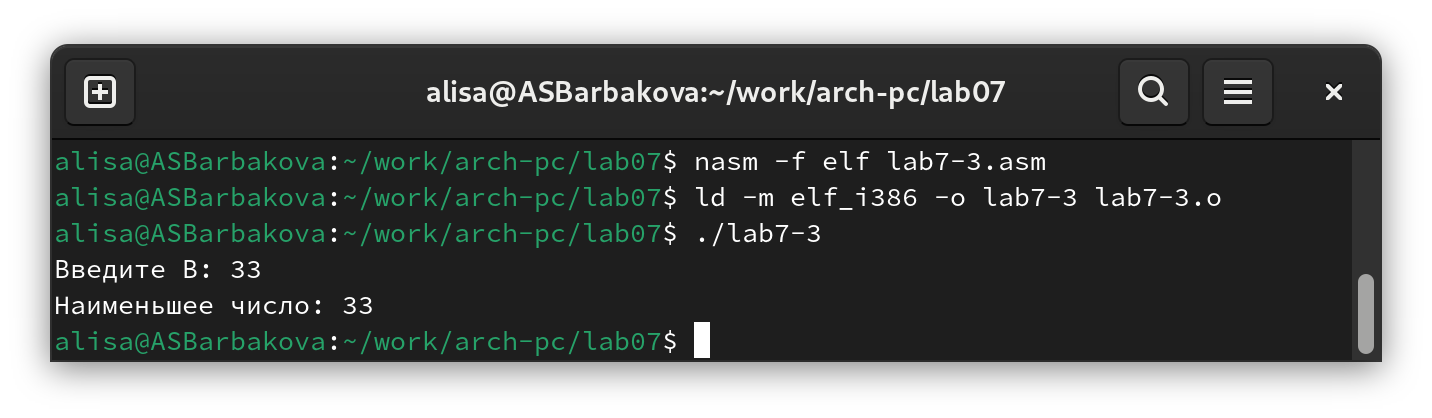


Рис. 15: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно 8-му варианту для введенных с клавиатурых переменных a и x (рис. -fig. 16).

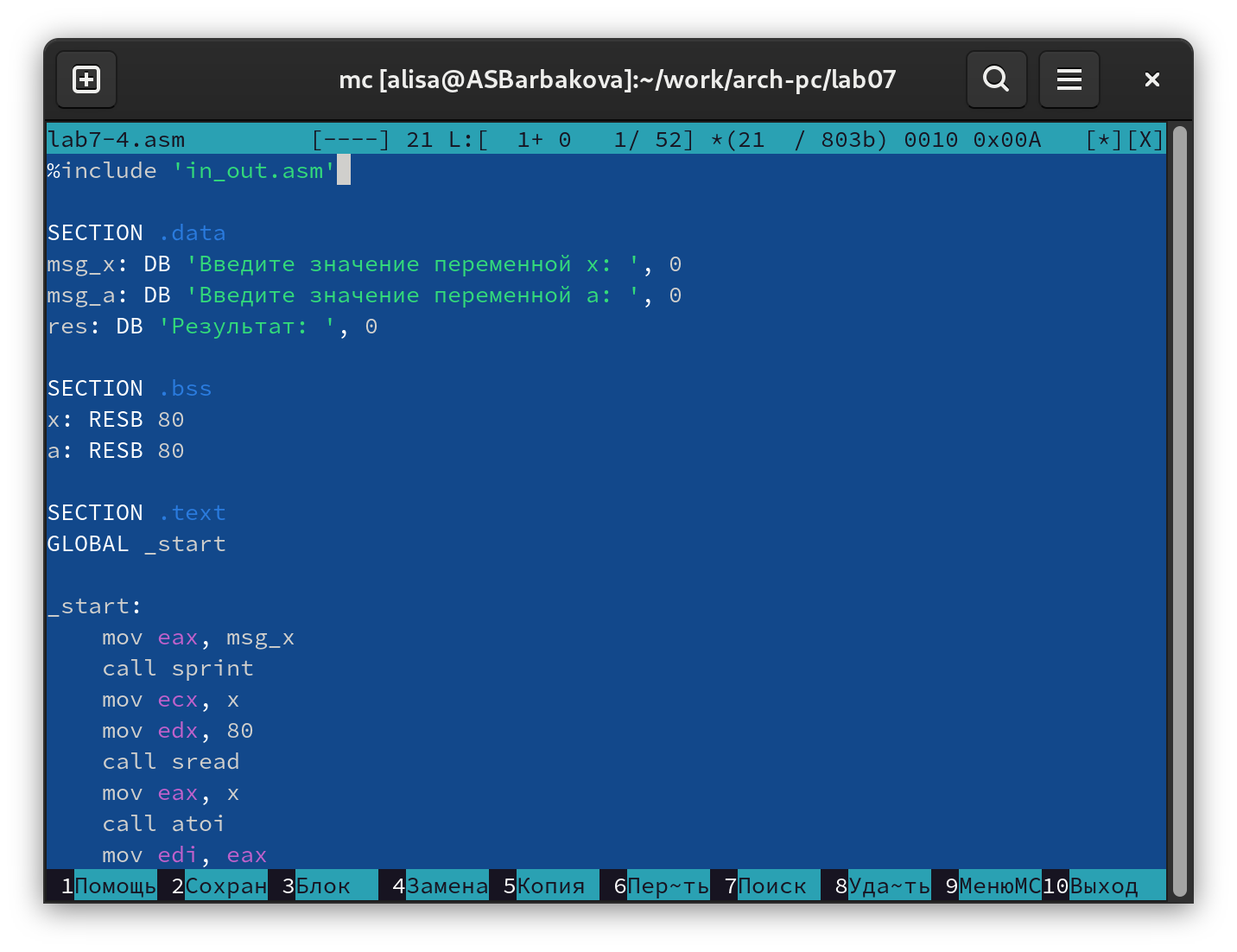


Рис. 16: Вторая программа самостоятельной работы

Код второй программы:

%include 'in\_out.asm'  
  
SECTION .data  
msg\_x: DB 'Введите значение переменной x: ', 0  
msg\_a: DB 'Введите значение переменной a: ', 0  
res: DB 'Результат: ', 0  
  
SECTION .bss  
x: RESB 80  
a: RESB 80  
  
SECTION .text  
GLOBAL \_start  
  
\_start:  
 mov eax, msg\_x  
 call sprint  
 mov ecx, x  
 mov edx, 80  
 call sread  
 mov eax, x  
 call atoi  
 mov edi, eax  
  
 mov eax, msg\_a  
 call sprint  
 mov ecx, a  
 mov edx, 80  
 call sread  
 mov eax, a  
 call atoi  
 mov esi, eax  
   
 cmp esi, 3  
 jl calculate\_three\_a  
  
 mov eax, edi  
 inc eax  
 jmp print\_result  
calculate\_three\_a:  
 mov eax, esi  
 imul eax, 3  
   
print\_result:   
 mov edi, eax  
 mov eax, res  
 call sprint  
 mov eax, edi  
 call iprintLF  
 call quit

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений a и x в соответствии с моим вариантом (рис. -fig. 17).

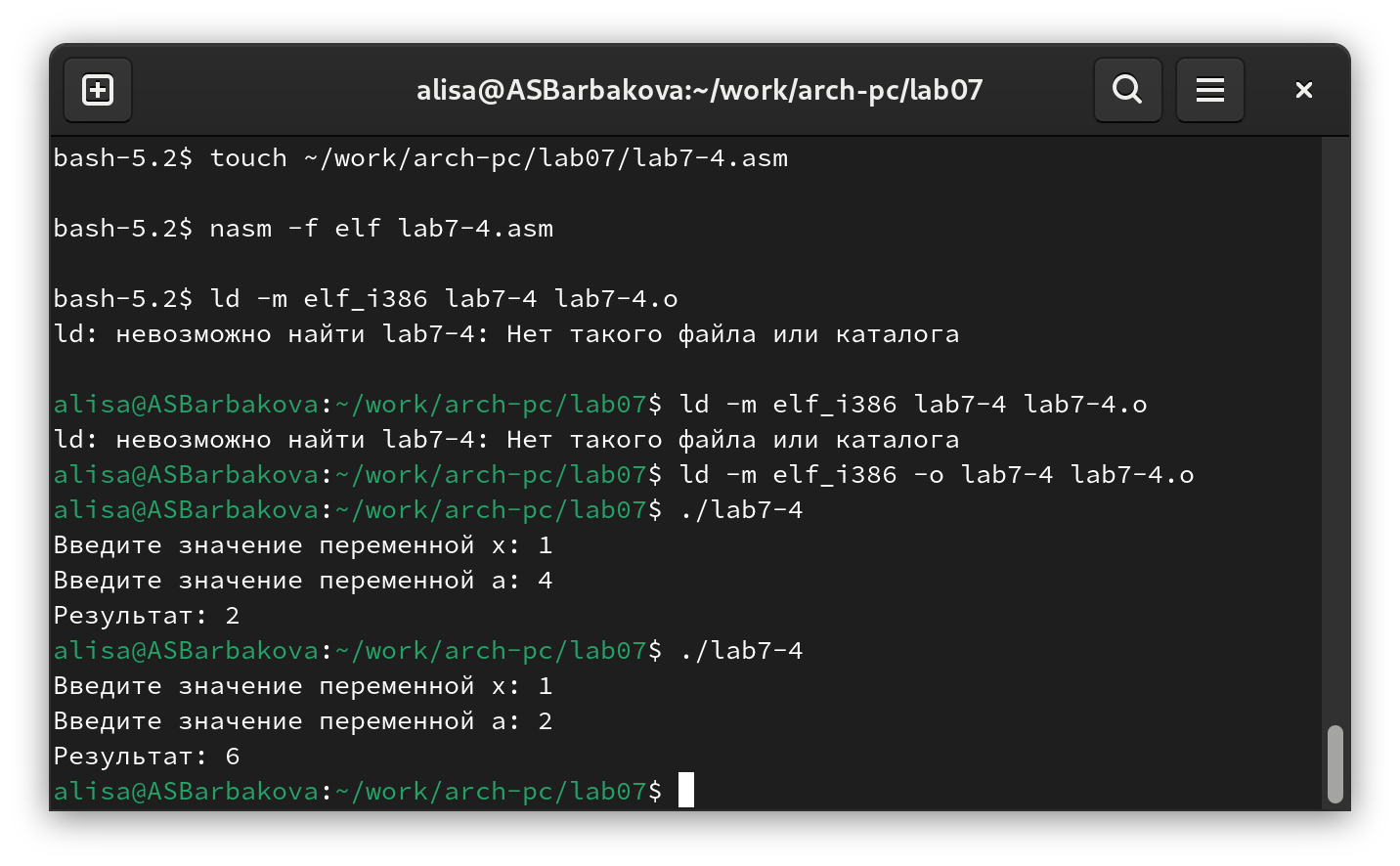


Рис. 17: Проверка работы второй программы

# 5 Выводы

При выполнении лабораторной работы я изучила команды условных и безусловных переходов, а также приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файлов листинга.

# Список литературы

1. [Лабораторная работа №7](https://esystem.rudn.ru/pluginfile.php/2089087/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%967.%20%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B%20%D0%B1%D0%B5%D0%B7%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B8%20%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%BF%D0%B5%D1%80%D0%B5%D1%85%D0%BE%D0%B4%D0%BE%D0%B2%20%D0%B2%20Nasm.%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%B2%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9..pdf)