Отчёт по лабораторной работе №6

Дисциплина: архитектура компьютера

Барбакова Алиса Саяновна

Содержание

# 1 Цель работы

Цель данной лабораторной работы - освоение арифметческих инструкций языка ассемблера NASM.

# 2 Задание

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

# 3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти.  
- Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: mov ax,bx.  
- Непосредственная адресация – значение операнда задается непосредственно в команде, Например: mov ax,2.  
- Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы.

# 4 Выполнение лабораторной работы

## 4.1 Символьные и численные данные в NASM

С помощью команды mkdir создаю директорию, в которой буду создавать файлы с программами. Перехожу в созданный каталог с помощью утилиты cd. С помощью утилиты touch создаю файл lab6-1.asm (рис. 1).

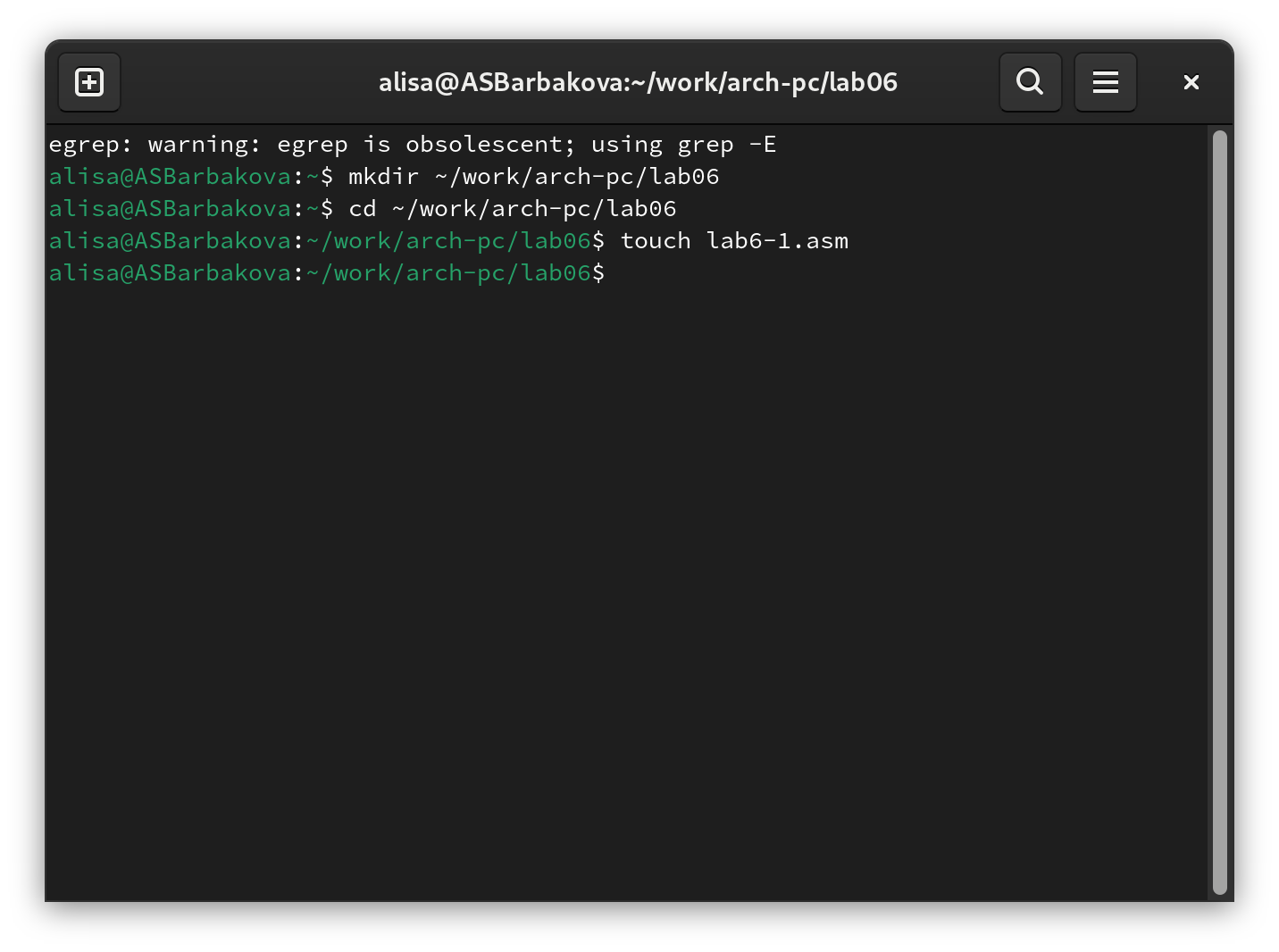


Рис. 1: Создание директории и файла

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

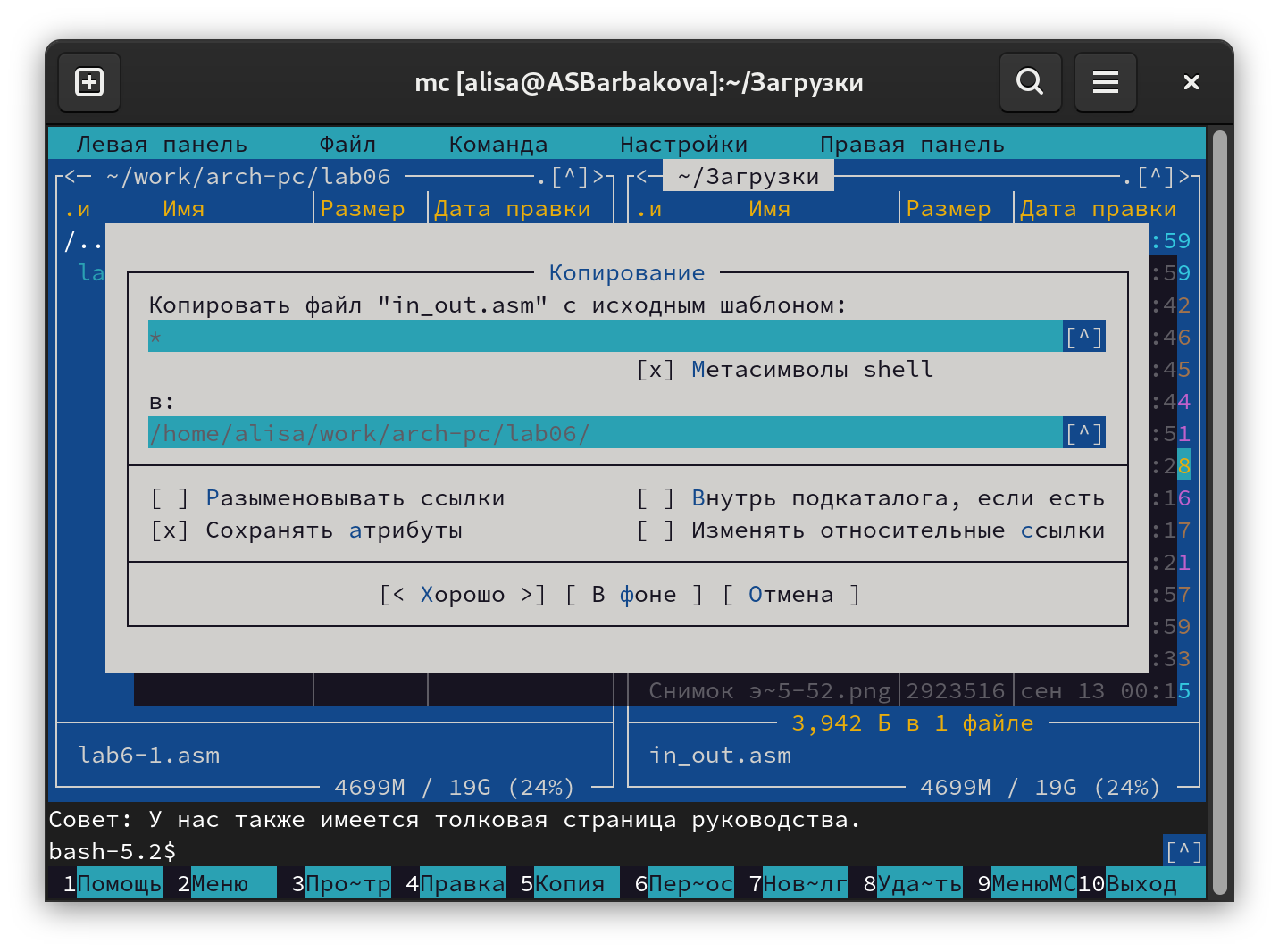


Рис. 2: Создание копии файла

Вставляю в созданный файл программу вывода значения регистра eax (рис. 3).

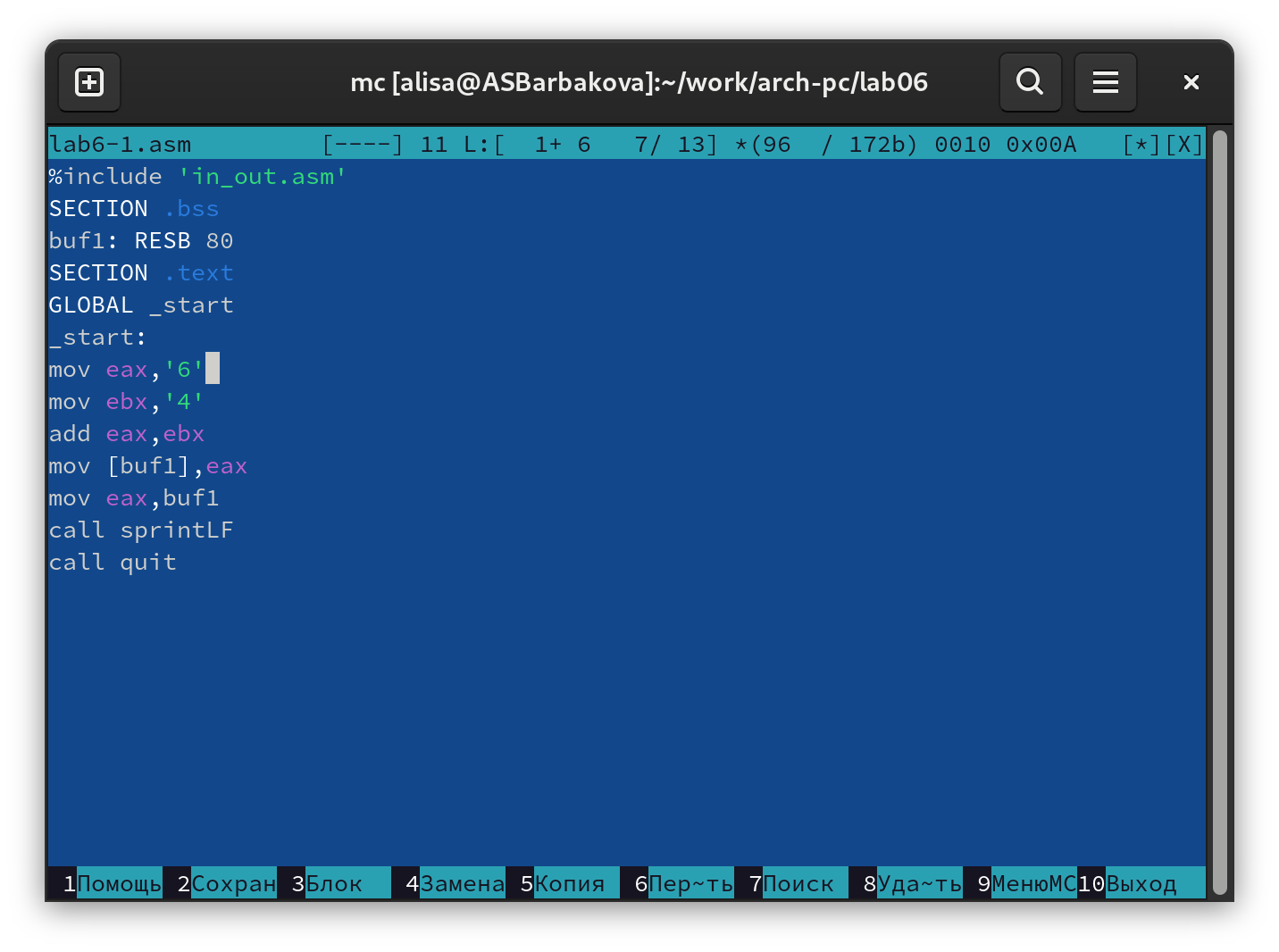


Рис. 3: Редактирование файла

Создаю исполняемый файл программы и запускаю его. Вывод программы: символ j (рис. 4).

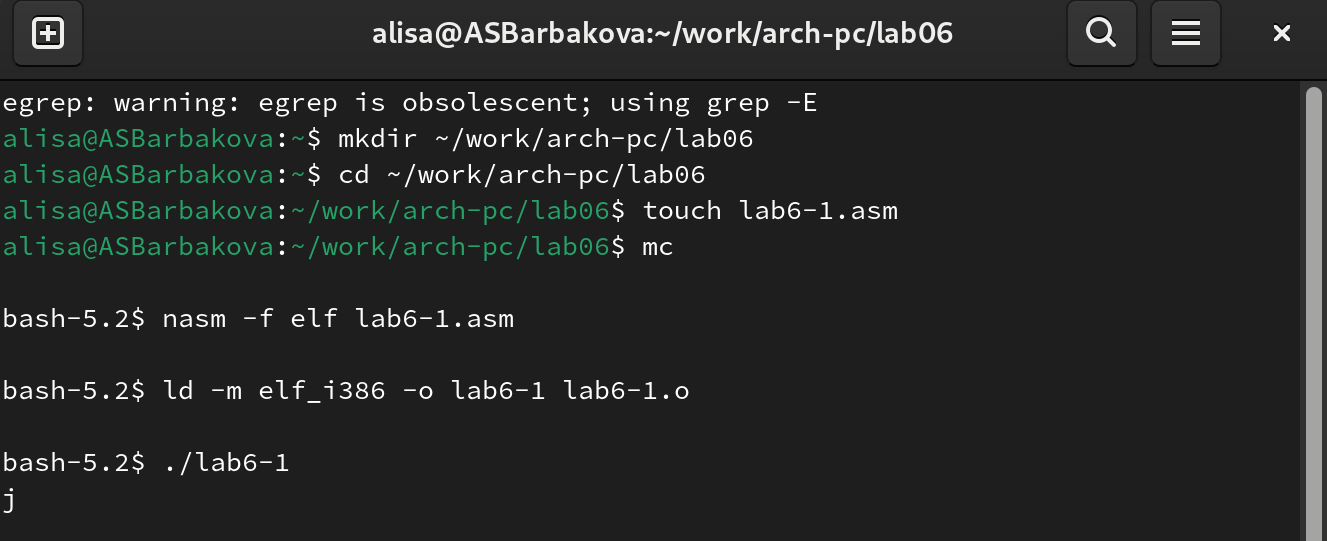


Рис. 4: Запуск исполняемого файла

Изменяю в тексте файла lab6-1.asm символы “6” и “4” на цифры 6 и 4 (рис. 5).

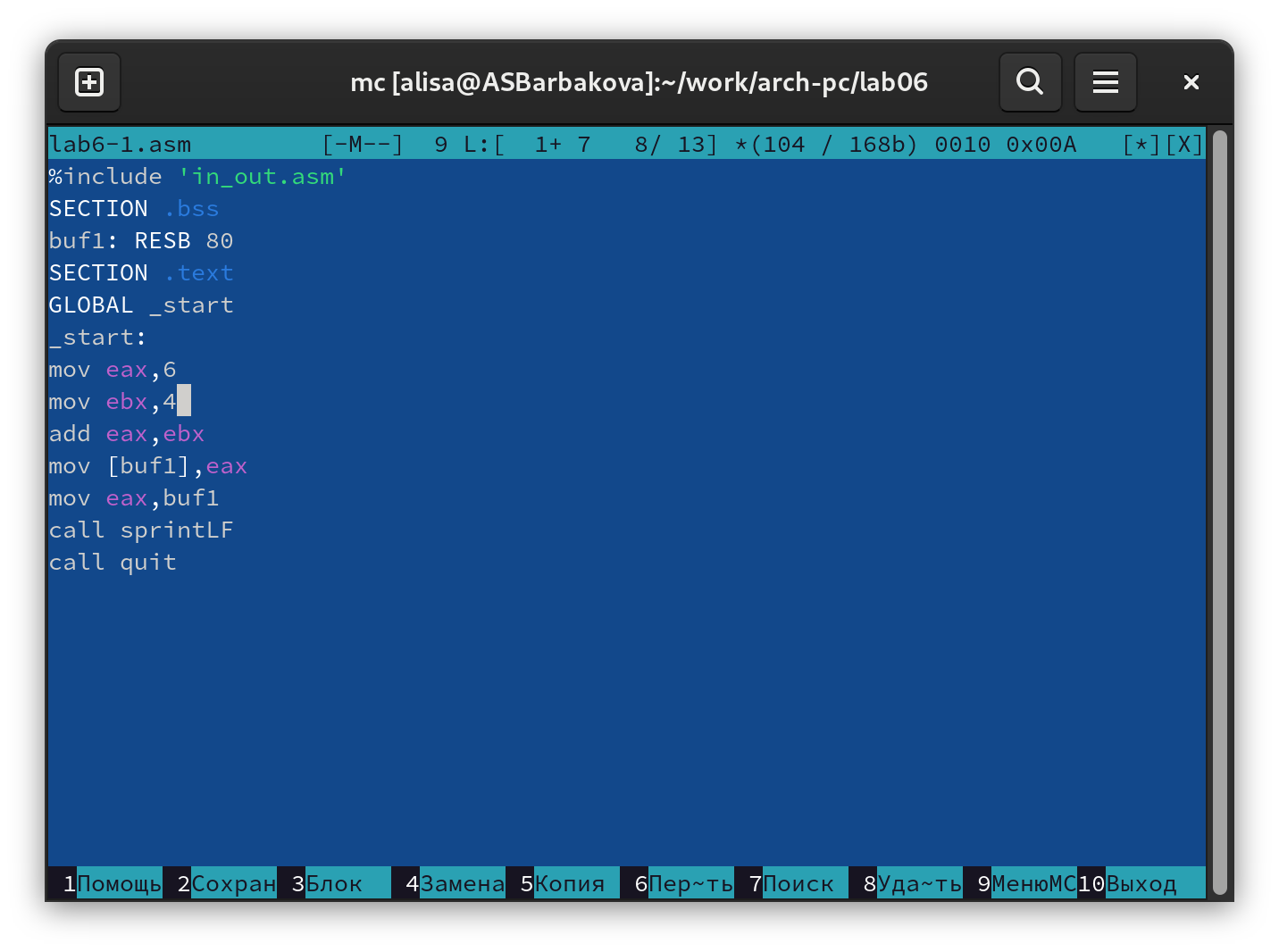


Рис. 5: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его. Этот символ не отображается при выводе на экран, так как является символом перевода строки (рис. 6).



Рис. 6: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью touch (рис. 7).

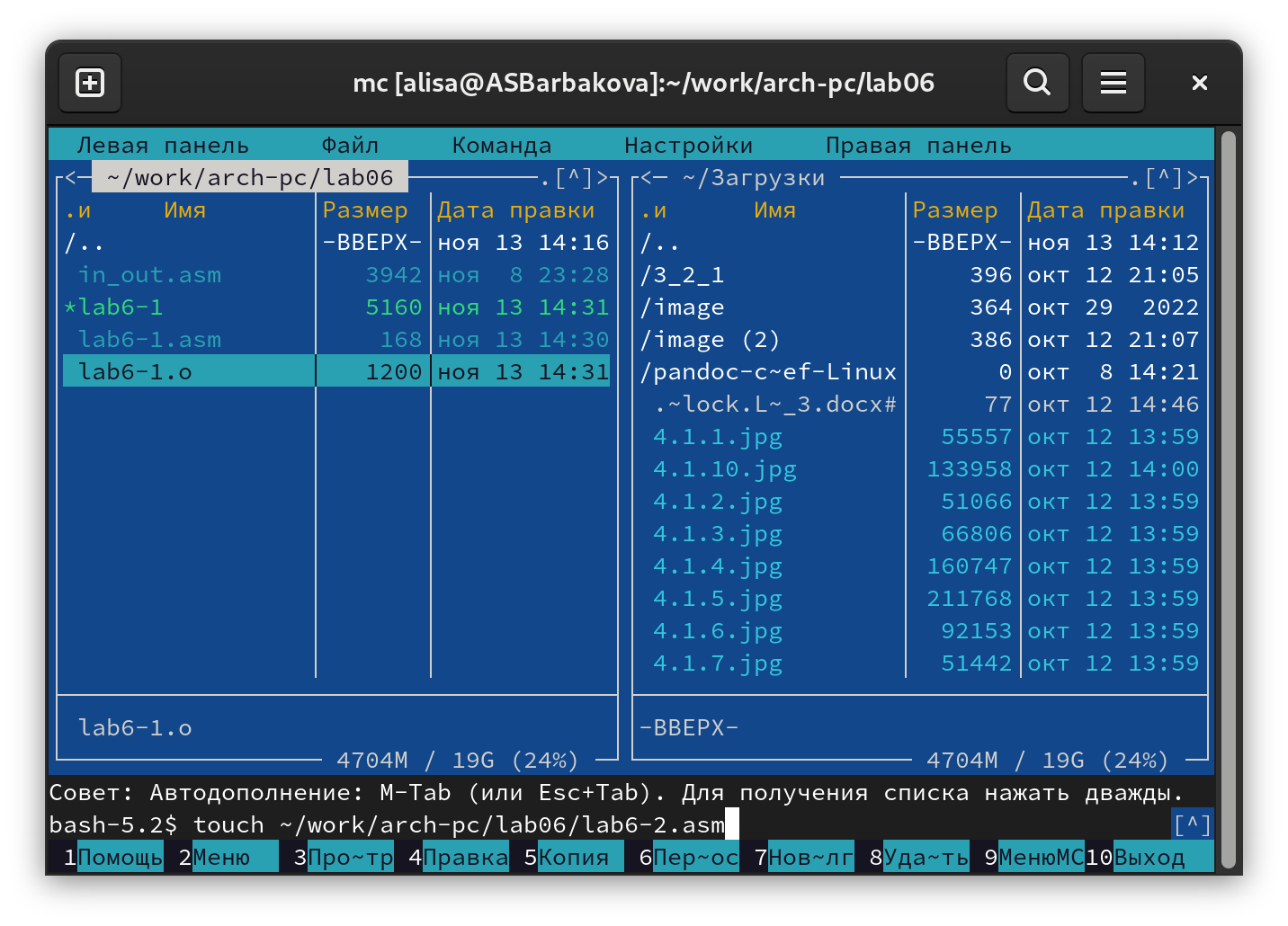


Рис. 7: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. 8).

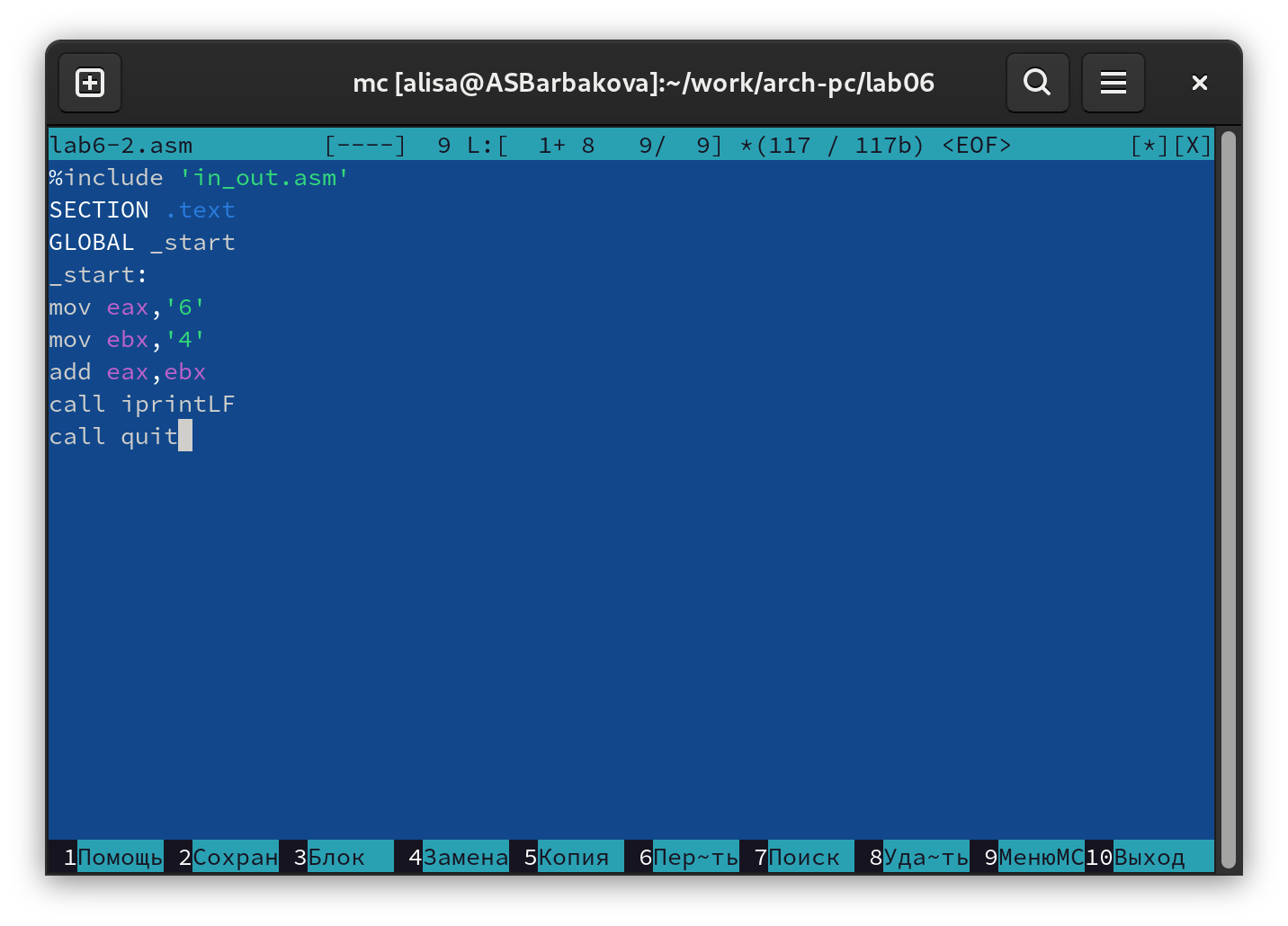


Рис. 8: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2. Выводится число 106 (рис. 9).



Рис. 9: Запуск исполняемого файла

Заменяю в тексте программы в файле символы “6” и “4” на числа 6 и 4 (рис. 10).

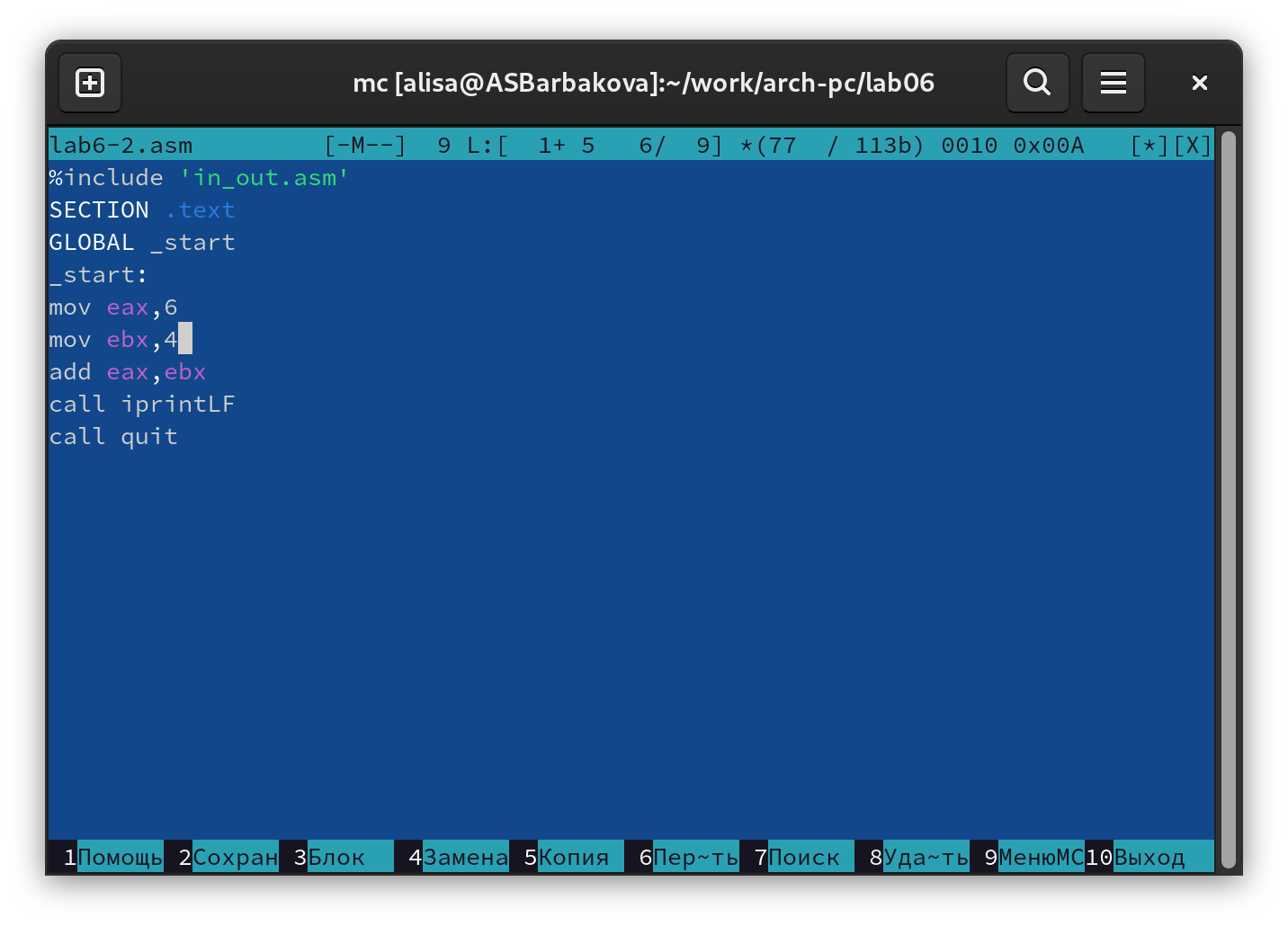


Рис. 10: Редактирование файла

Создаю и запускаю новый исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10 (рис. 11).

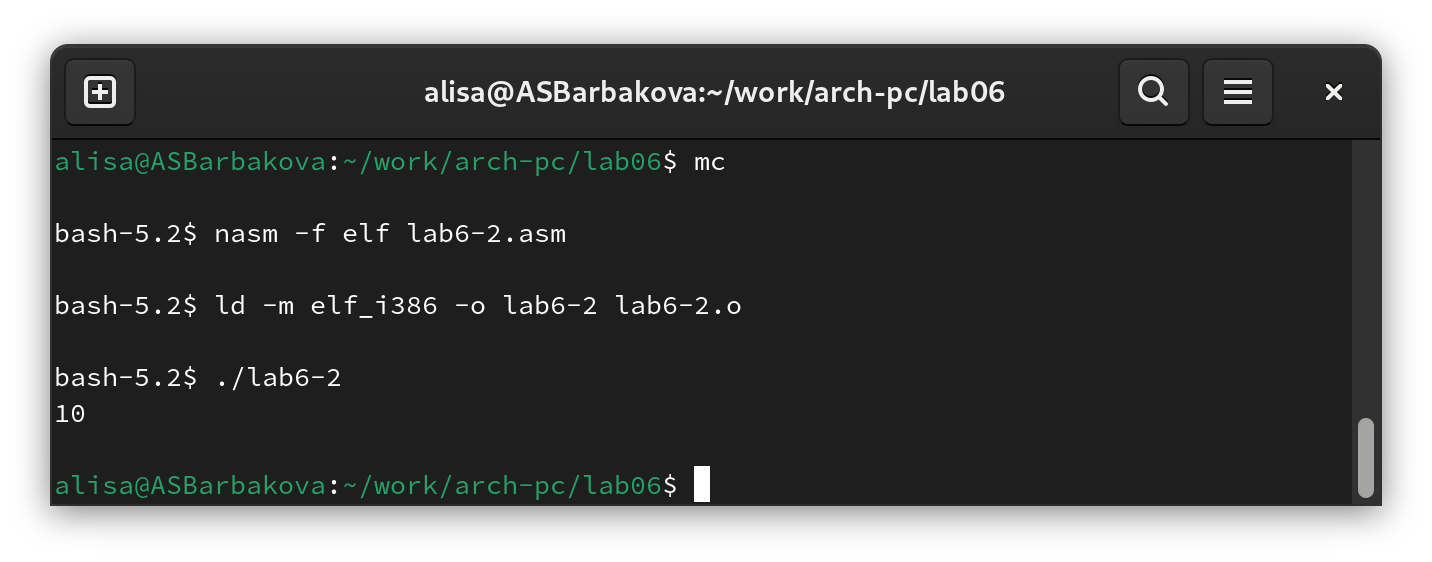


Рис. 11: Запуск исполняемого файла

Заменяю в тексте программы файла lab6-2.asm функцию iprintLF на iprint (рис. 12).

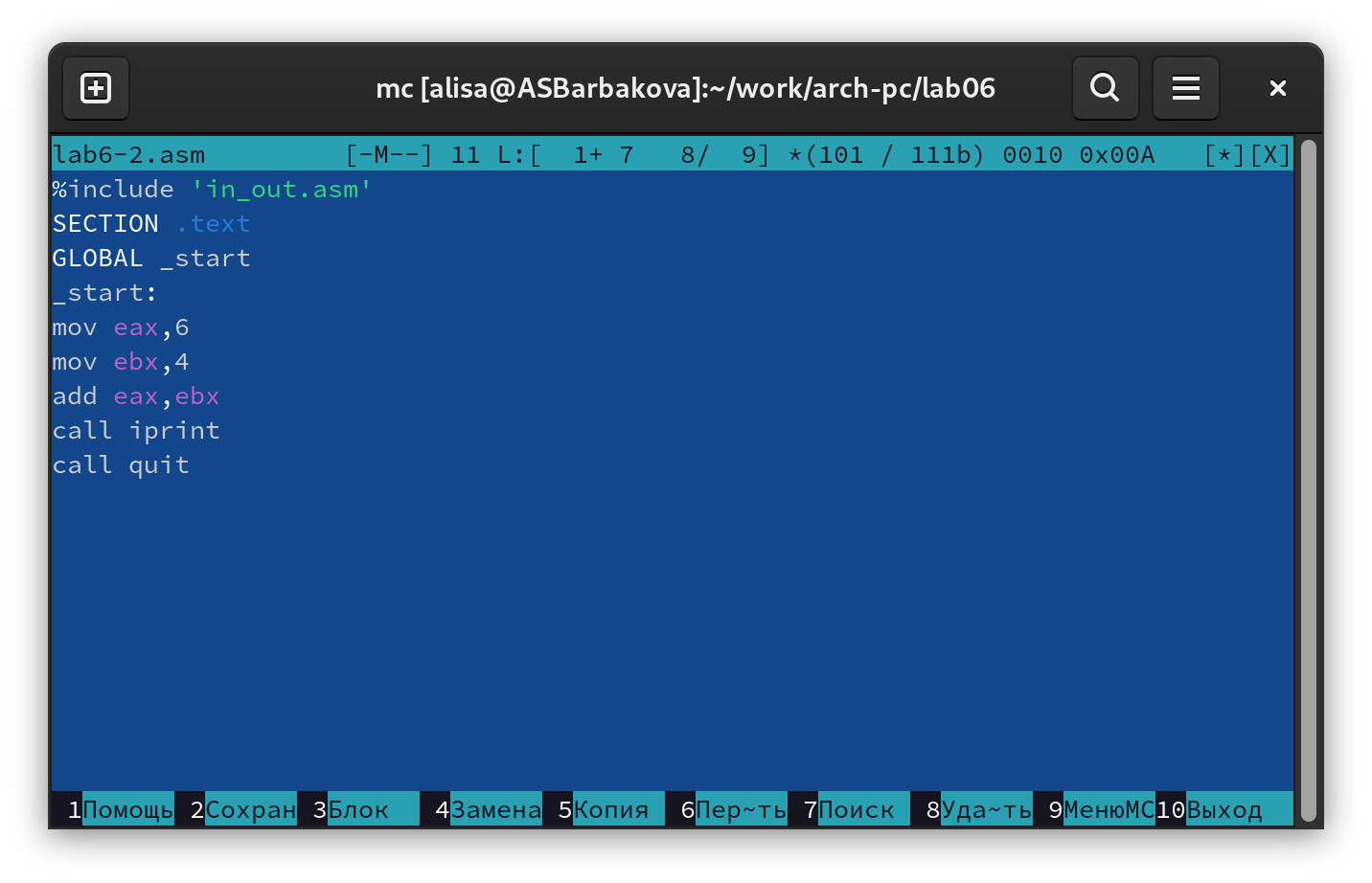


Рис. 12: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 13). Вывод не изменился, потому что символ переноса строки не отображался, когда программа исполнялась с функцией iprintLF, а iprint не добавляет к выводу символ переноса строки.

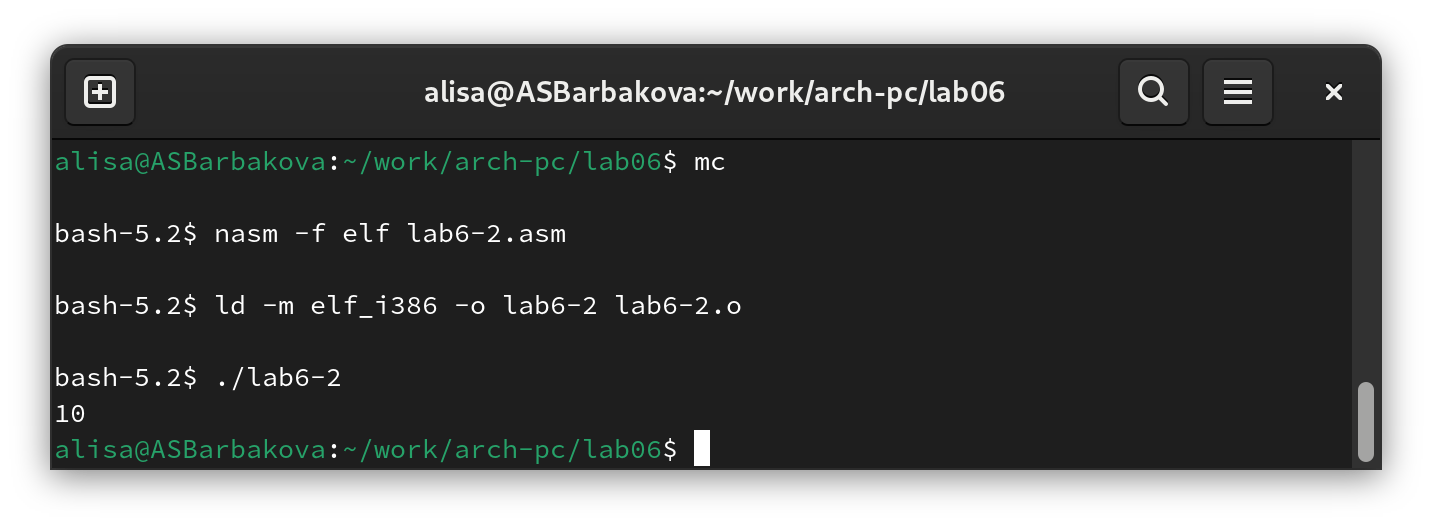


Рис. 13: Запуск исполняемого файла

## 4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch (рис. 14).

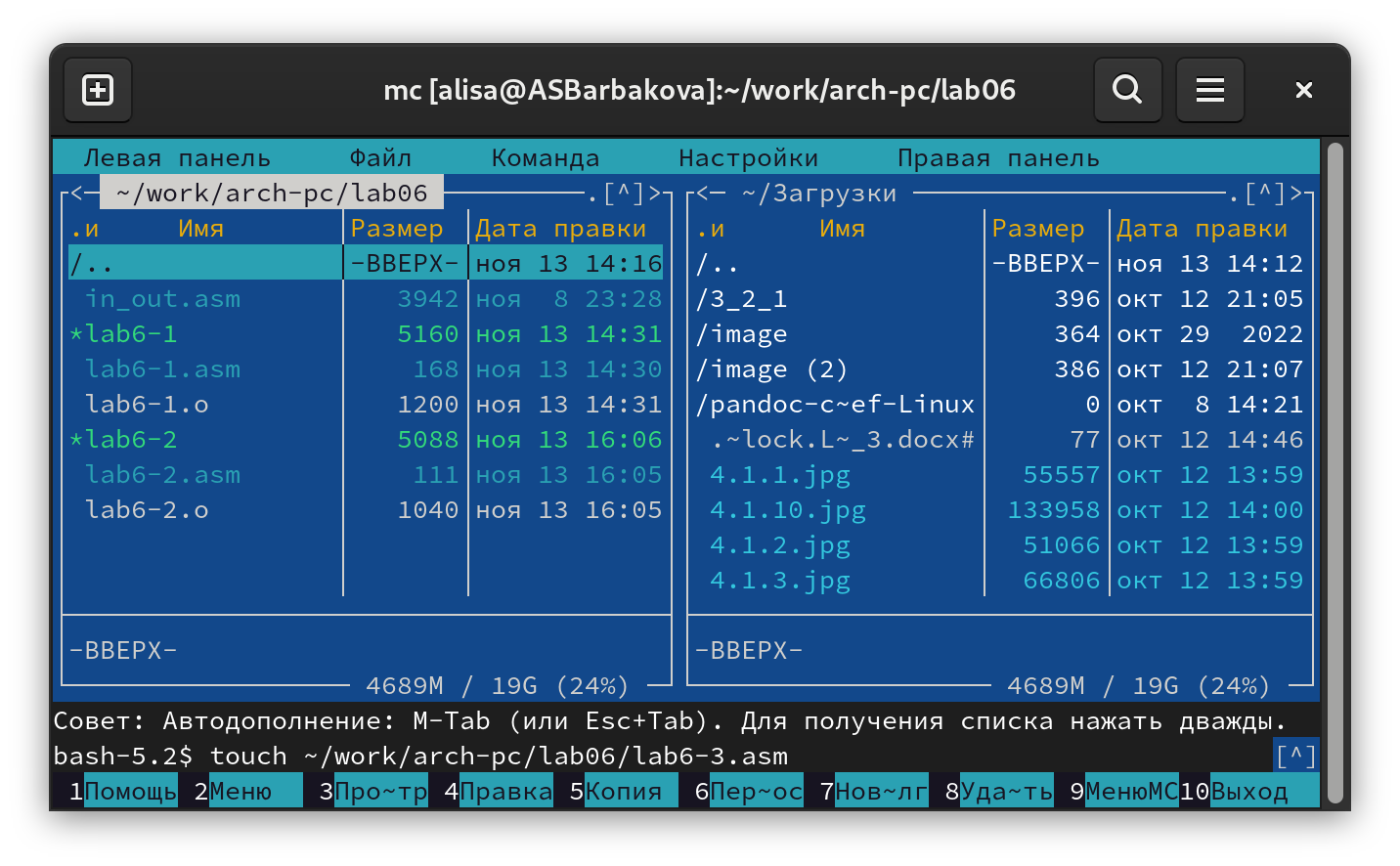


Рис. 14: Создание файла

Ввожу в созданный файл текст программы для вычисления f(x) = (5 \* 2 + 3)/3 (рис. 15).

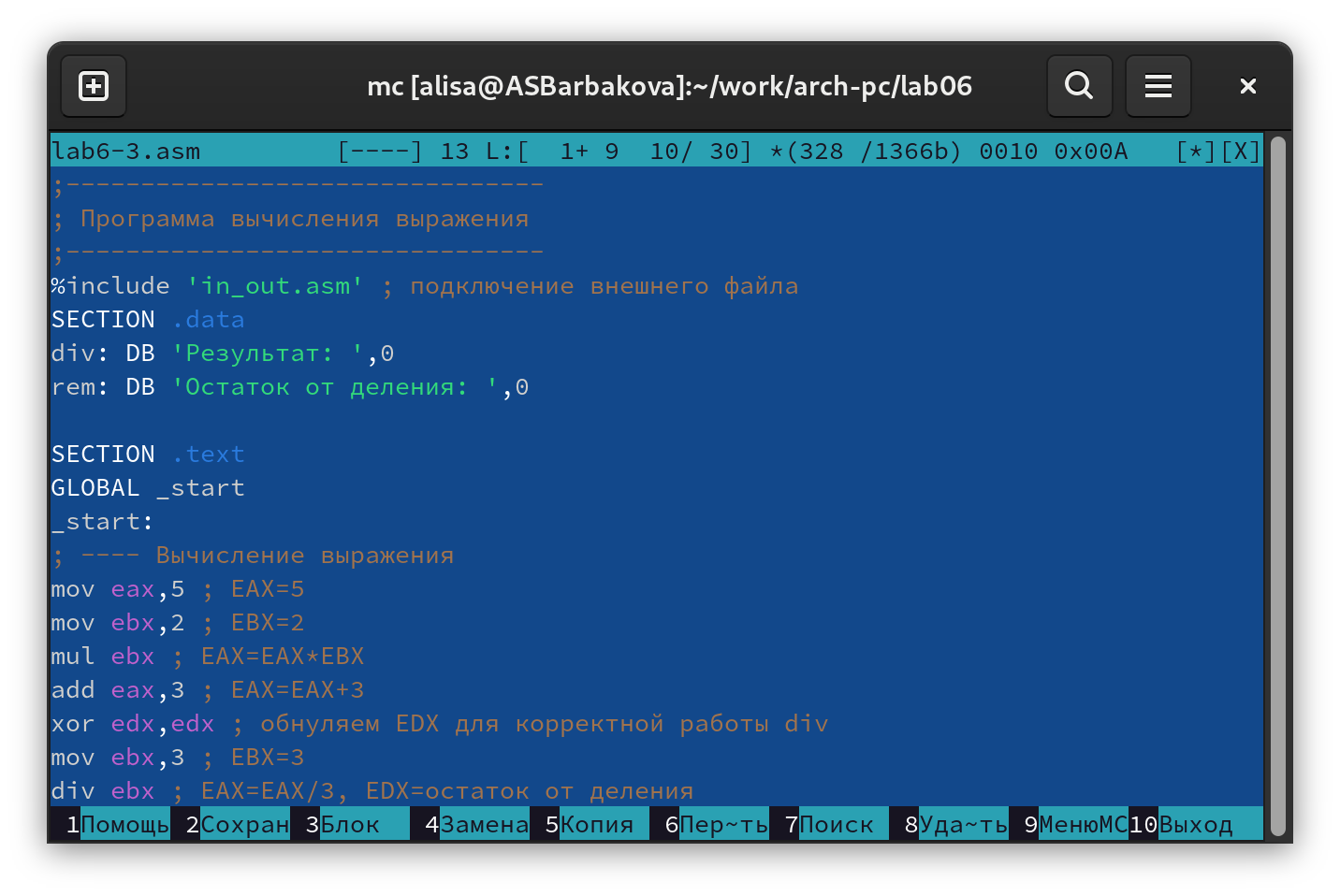


Рис. 15: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 16). Решение программы совпадает с ответом.



Рис. 16: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения f(x) = (4 \* 6 + 2)/5 (рис. 17).

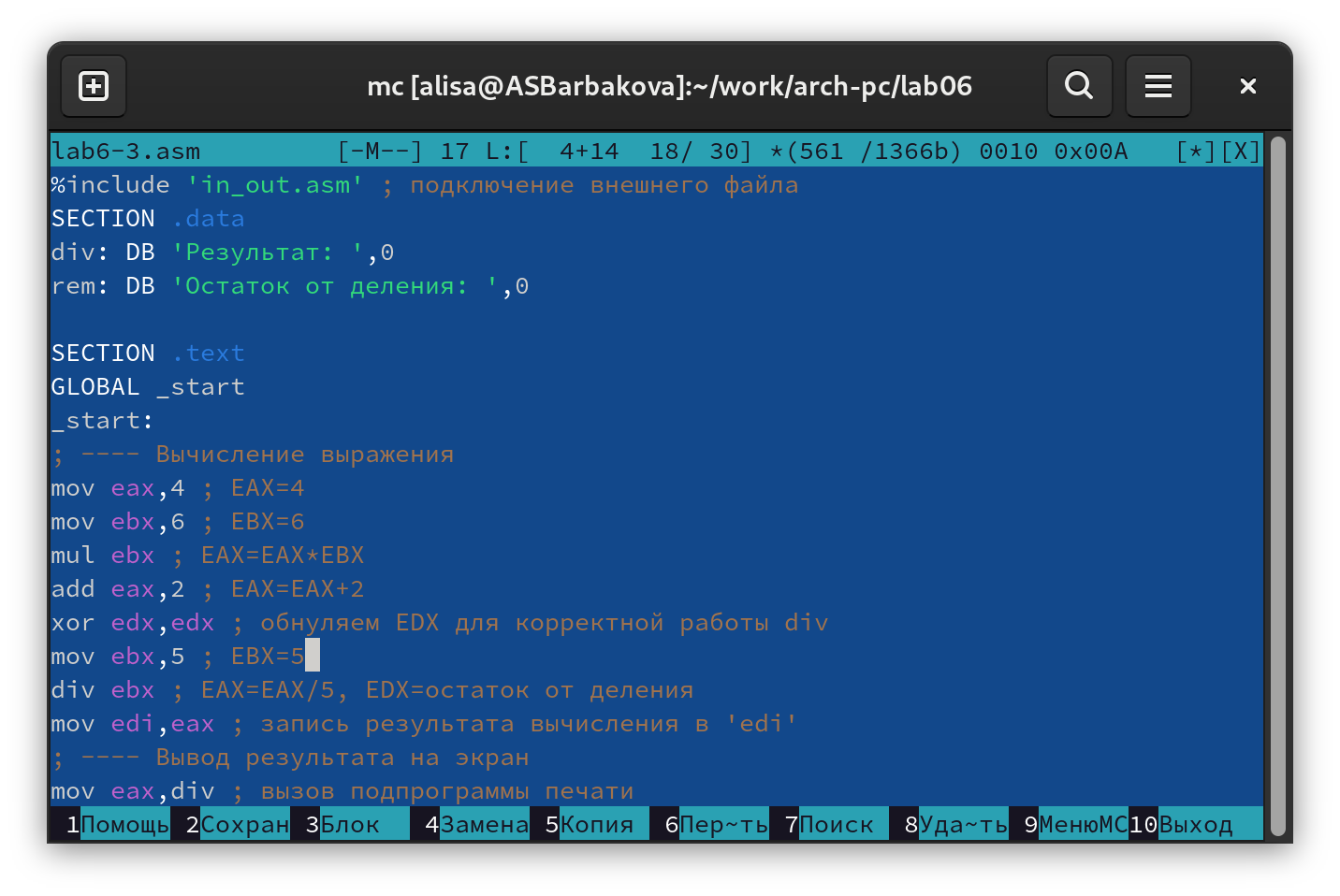


Рис. 17: Изменение программы

Создаю и запускаю новый исполняемый файл (рис. 18). Программа сработала верно.



Рис. 18: Запуск исполняемого файла

Создаю файл variant.asm с помощью команды touch (рис. 19).

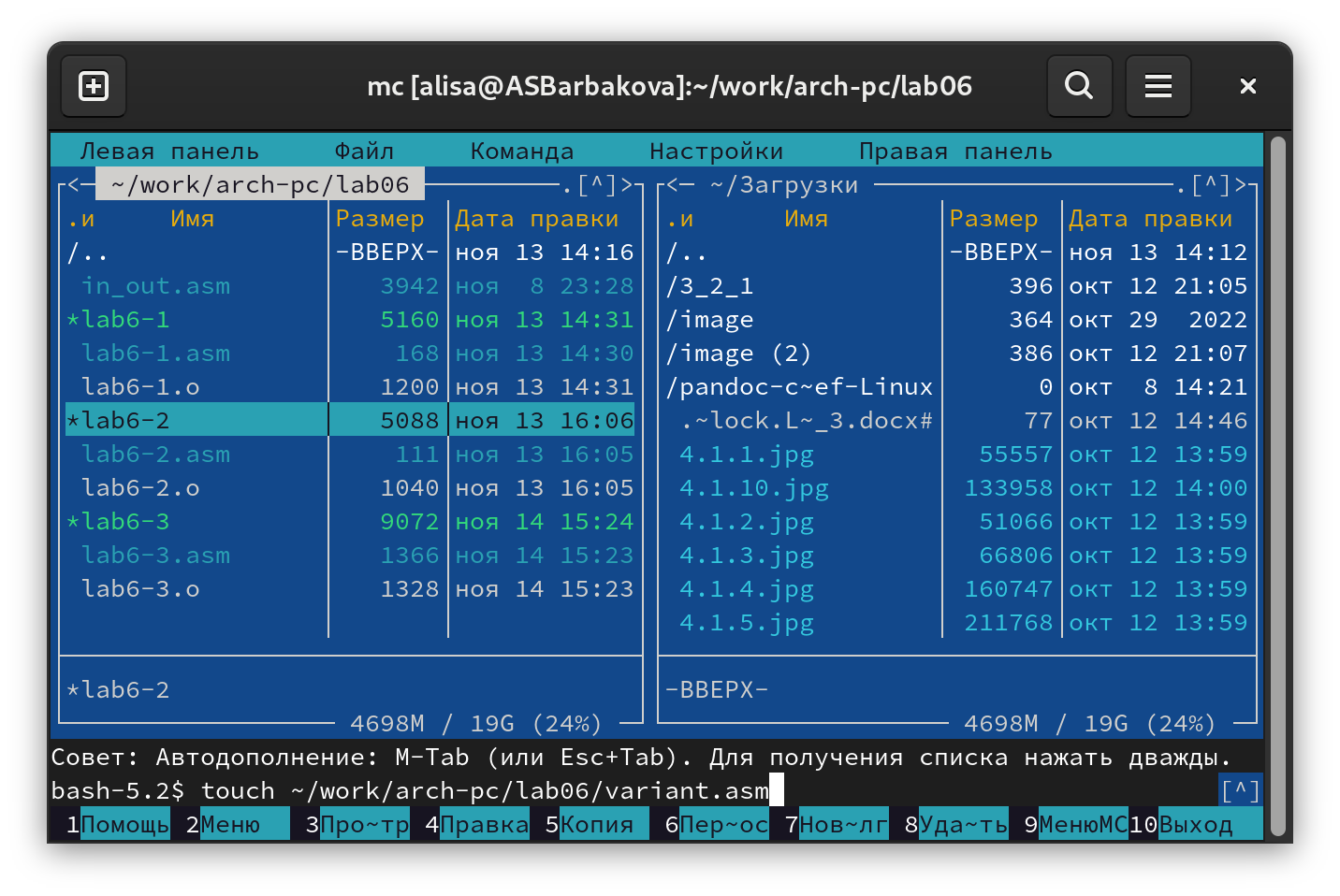


Рис. 19: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру моего студенческого билета (рис. 20).

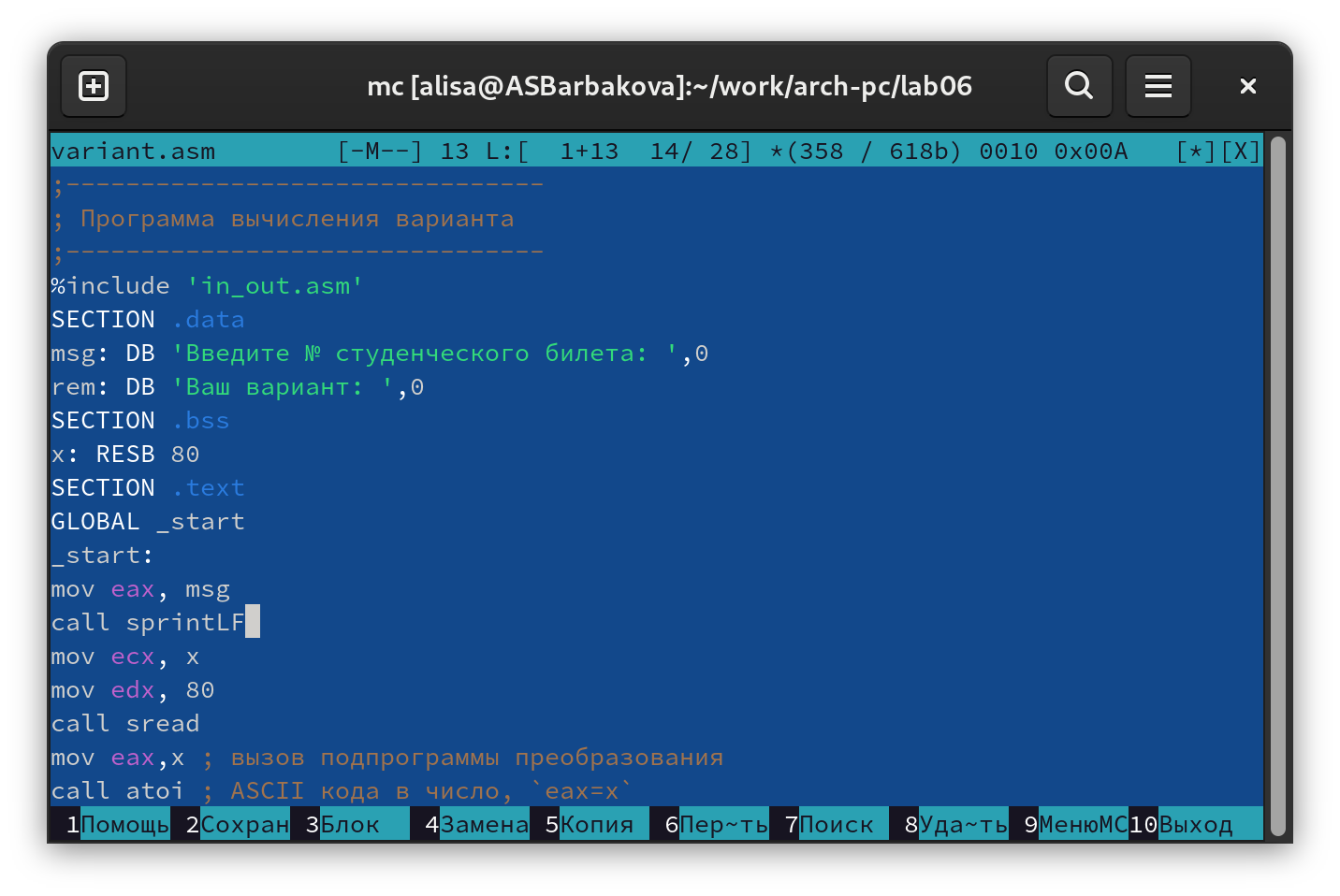


Рис. 20: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 21). Ввожу номер своего студ. билета с клавиатуры. Программа выводит, что мой вариант - 8.

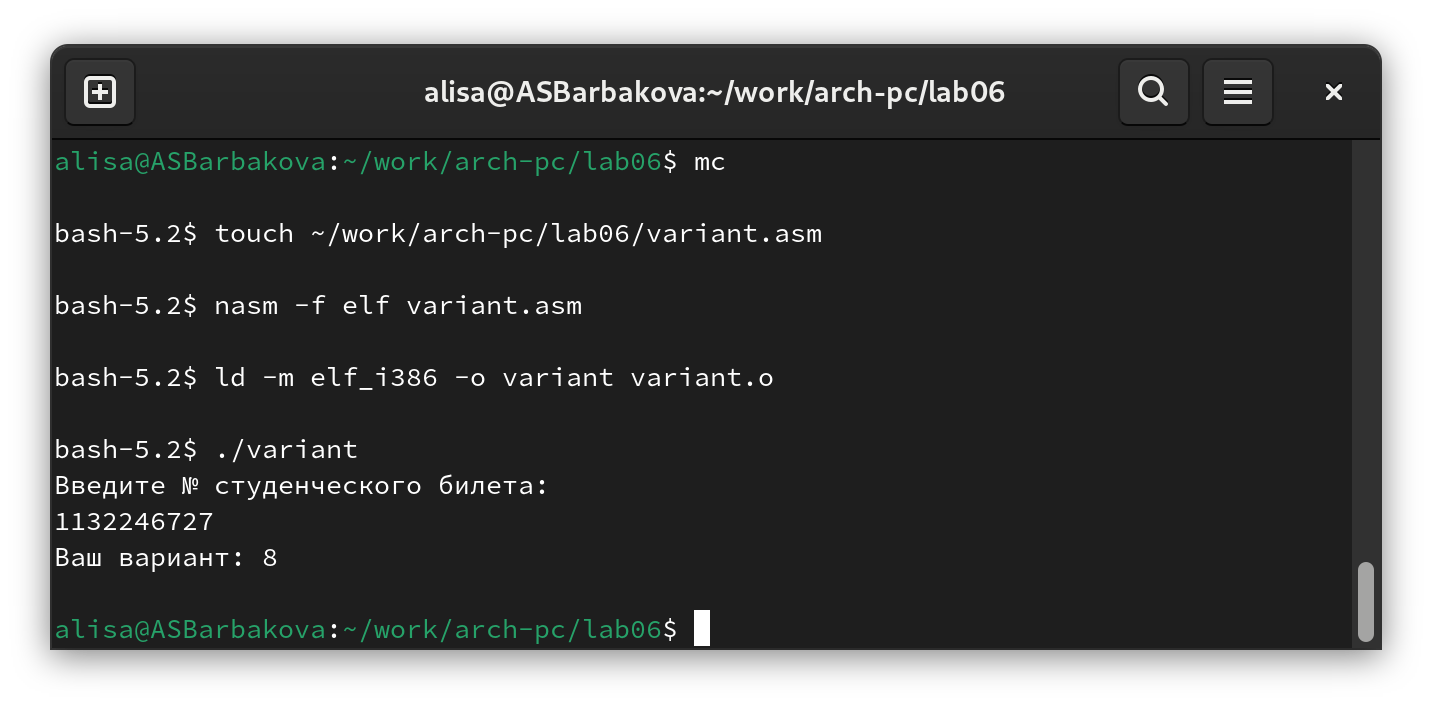


Рис. 21: Запуск исполняемого файла

### 4.2.1 Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

mov eax,rem  
call sprint

1. Инструкция mov ecx, x используется, чтобы положить адрес вводимой строки x в регистр ecx mov edx, 80 - запись в регистр edx длины вводимой строки call sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры
2. Инструкция call atoi вызывает подпрограмму atoi из внешнего файла, которая преобразует строку ASCII, находящуюся в x, в целое число и записывает его в регистр eax.
3. За вычисления варианта отвечают строки:

xor edx,edx  
mov ebx,20   
div ebx   
inc edx

1. Остаток от деления записывается в регистр edx при div ebx.
2. Инструкция inc edx увеличивает значение в регистре edx на 1.
3. За вывод на экран результатов вычислений отвечают строки:

mov eax,edx  
call iprintLF

## 4.3 Выполнение заданий для самостоятельной работы

Создаю файл task.asm с помощью утилиты touch. Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления выражение под вариантом 8: (11 + x) \* 2 - 6 (рис. 22).

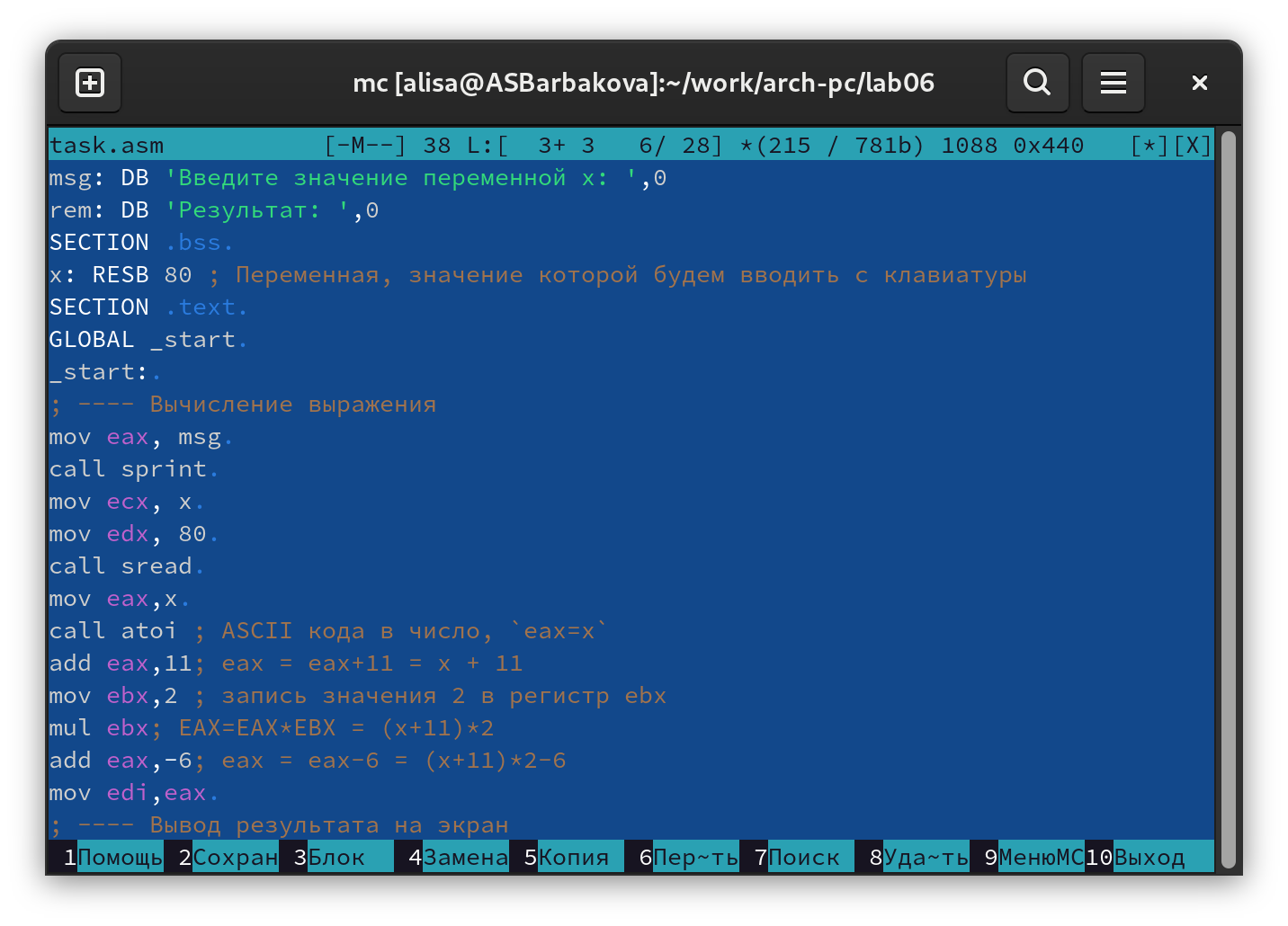


Рис. 22: Написание программы

Создаю исполняемый файл (рис. 23).

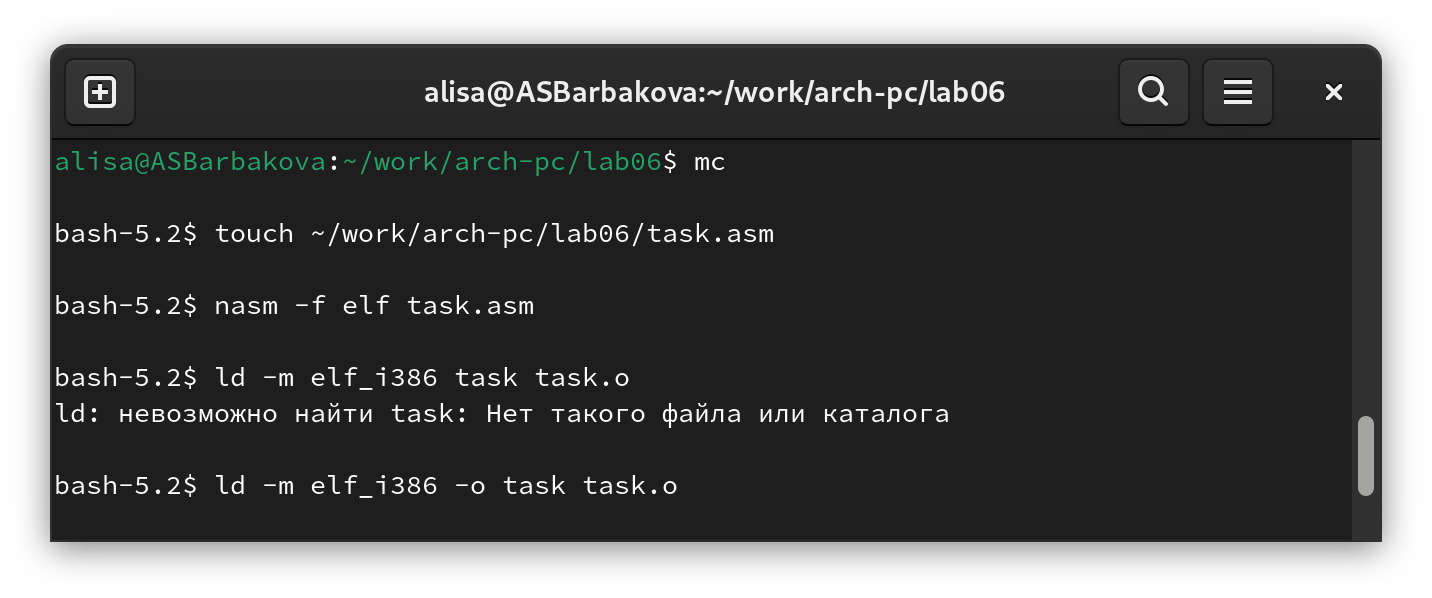


Рис. 23: Создание исполняемого файла

Запускаю исполняемый файл, ввожу x1=1 (рис. 24).

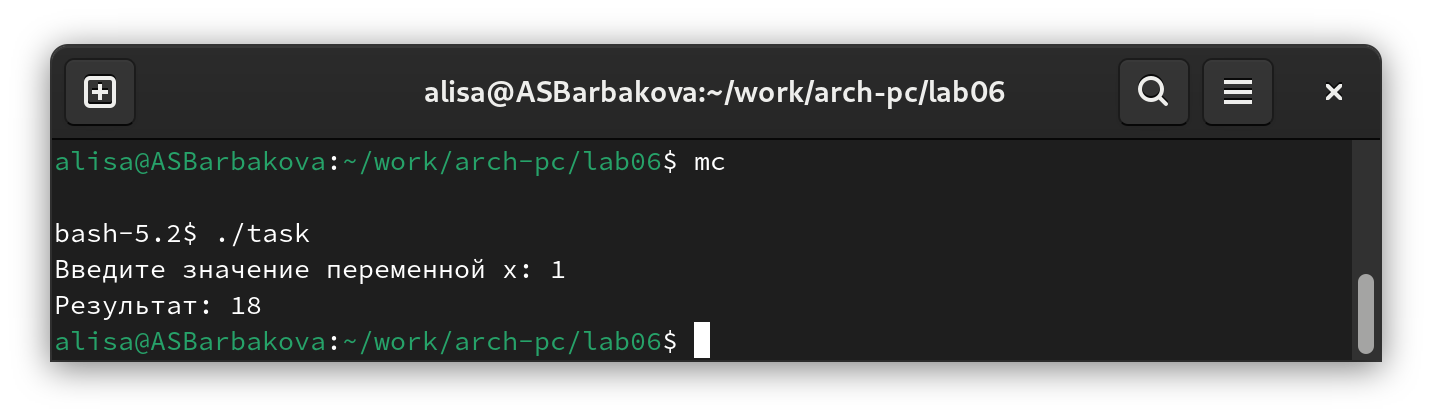


Рис. 24: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла, ввожу x2=9 (рис. 25). Программа сработала верно.

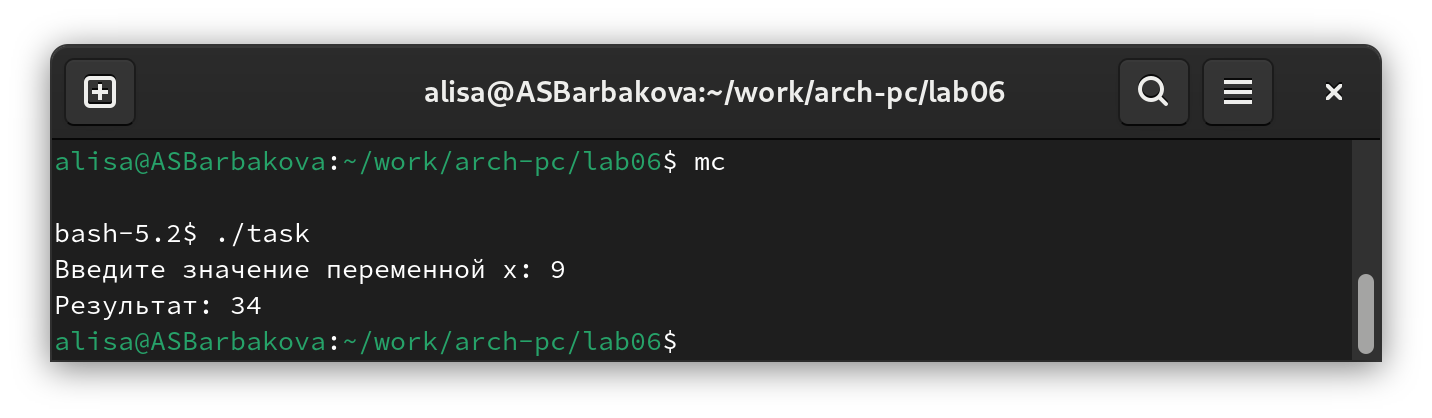


Рис. 25: Запуск исполняемого файла

**Программа для вычисления значения выражения (11 + x) \* 2 − 6.**

%include 'in\_out.asm'   
SECTION .data   
msg: DB 'Введите значение переменной х: ',0  
rem: DB 'Результат: ',0  
SECTION .bss   
x: RESB 80 ; Переменная, значение которой будем вводить с клавиатуры  
SECTION .text   
GLOBAL \_start   
\_start:   
; ---- Вычисление выражения  
mov eax, msg   
call sprint   
mov ecx, x   
mov edx, 80   
call sread   
mov eax,x   
call atoi ; ASCII кода в число, `eax=x`  
add eax,11; eax = eax+11 = x + 11  
mov ebx,2 ; запись значения 2 в регистр ebx  
mul ebx; EAX=EAX\*EBX = (x+11)\*2  
add eax,-6; eax = eax-6 = (x+11)\*2-6  
mov edi,eax   
; ---- Вывод результата на экран  
mov eax,rem   
call sprint   
mov eax,edi   
call iprint  
call quit

# 5 Выводы

При выполнении данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

# Список литературы

1. [Лабораторная работа №6](https://esystem.rudn.ru/pluginfile.php/2089086/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%966.%20%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5%20%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8%20%D0%B2%20NASM..pdf)