Отчет по выполнению лабораторной работы №6

Дисциплина: архитектура компьютеров

Новиков Никита Владимирович

Содержание

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

# 2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM

# 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. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно

# 4 Выполнение лабораторной работы

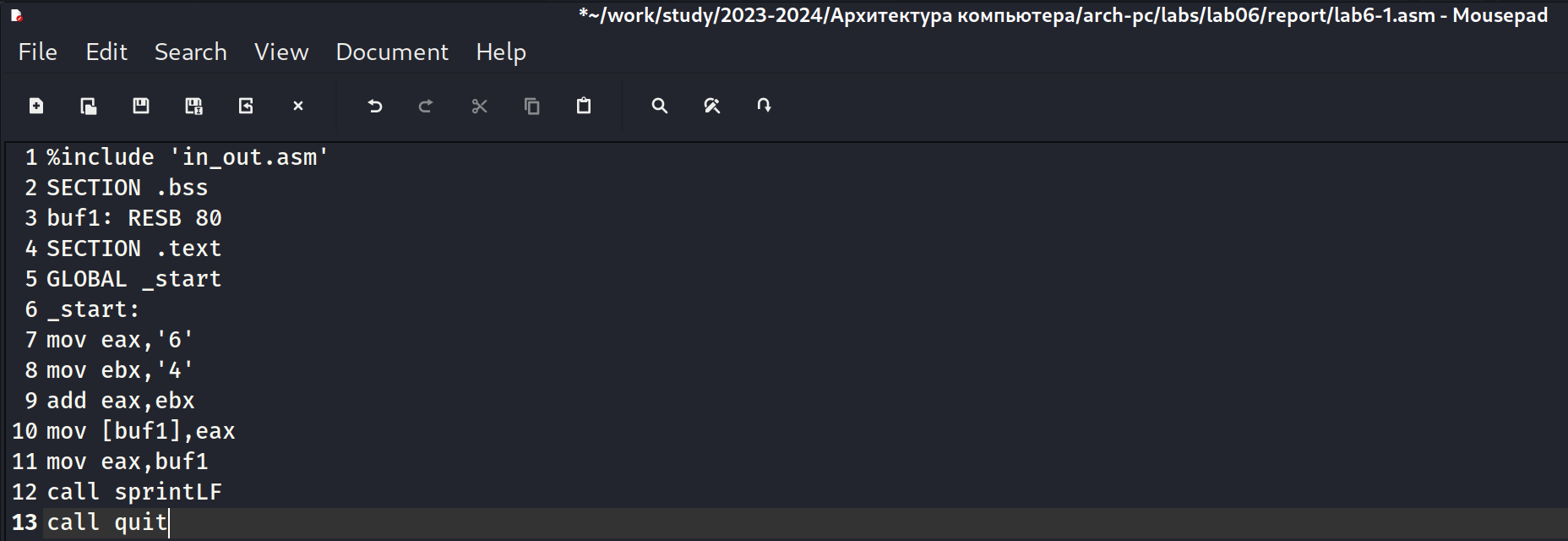
## 4.1 Символьные и численные данные в NASM

Создаю каталог для программ лабораторной работы №6, перехожу в него и создаю файл lab6-1.asm (рис. [??]).



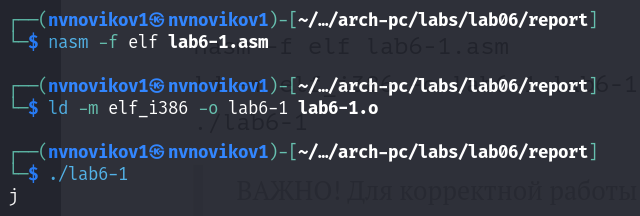
Создание каталога

Ввожу в файл lab6-1.asm текст программы из листинга 6.1.(рис. [??]).



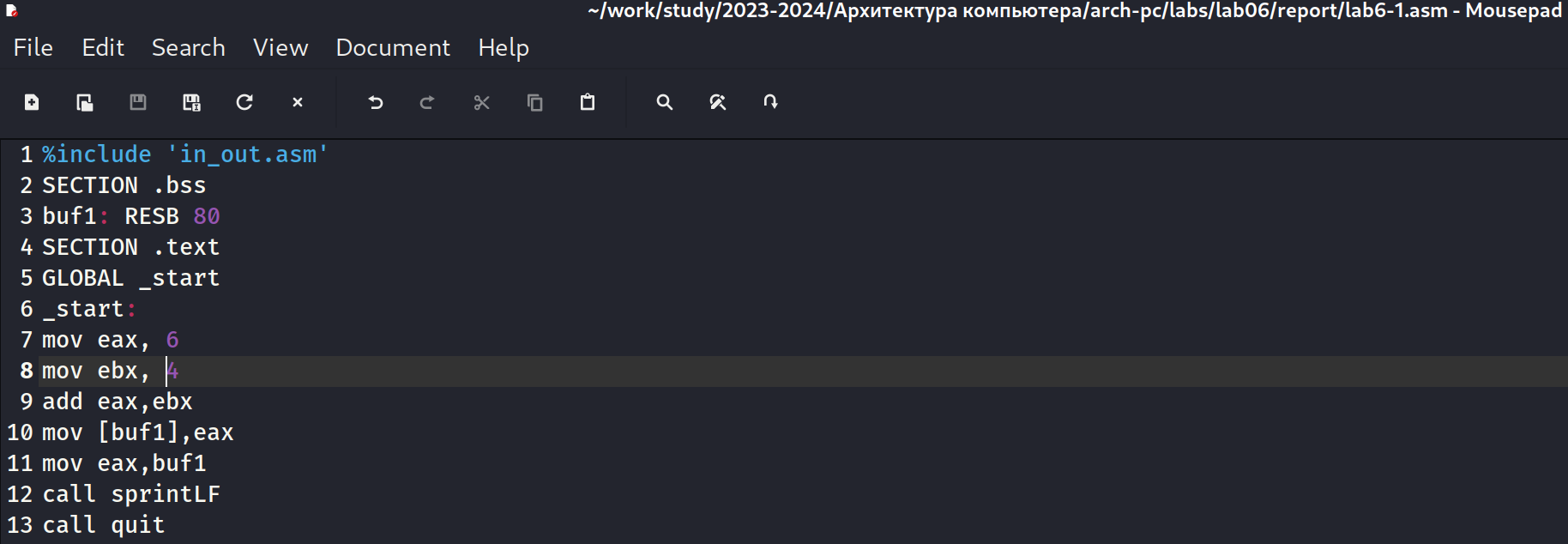
Ввод программы из листинга 6.1

Создаю исполняемый файл и запускаю его.(рис. [??]).



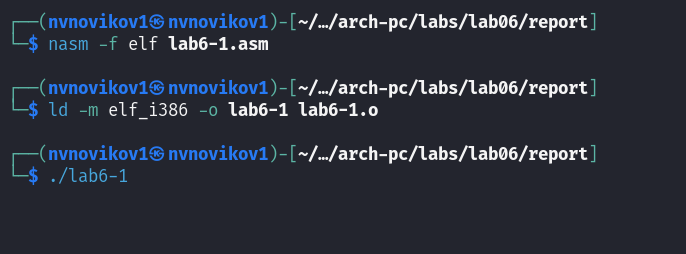
Запуск файла

Изменяю текст программы и вместо символов записываю в регистры числа.(рис. [??]).



Исправление программы

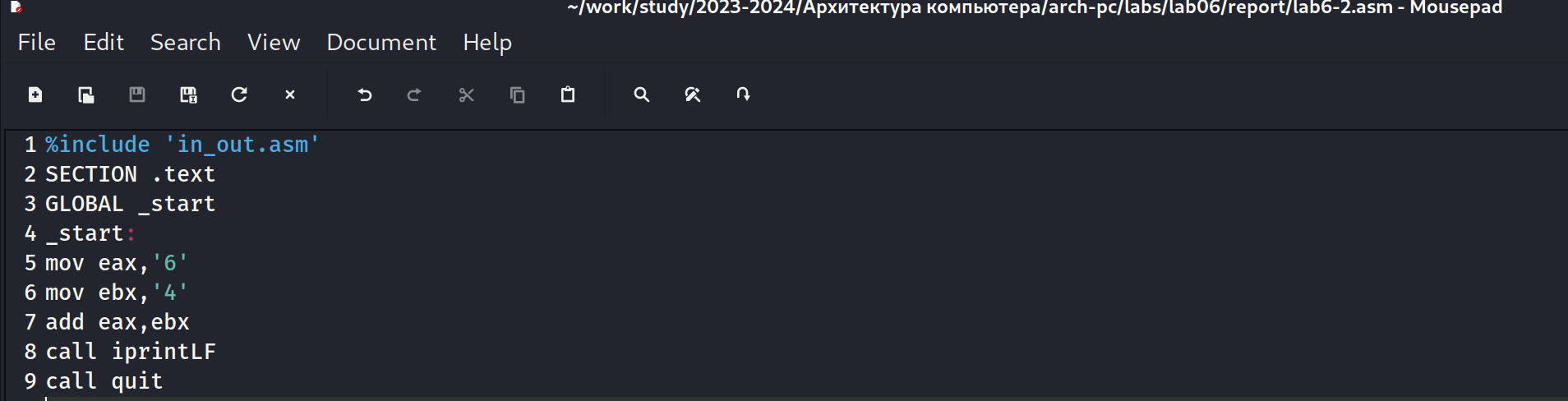
Далее создаю исполняемый файл и запускаю его.(рис. [??]).



Запуск исправленной программы

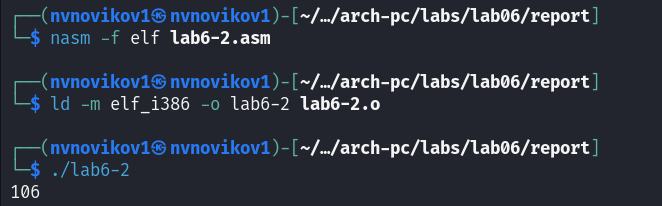
Пользуясь таблицей ASCII можно определить,что код 10 соответствует символу переносу строки. Этот символ не отображается на экране.

Создаю новый файл lab6-2 в том же каталоге и ввожу в него текст программы из листинга 6.2.(рис. [??]).



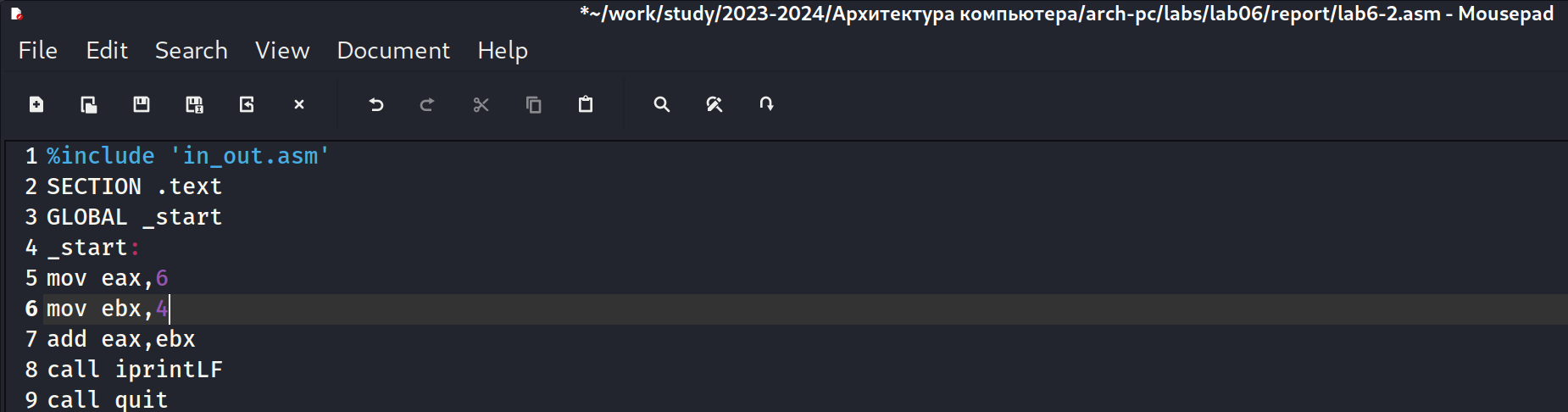
Создание нового файла

Создаю исполняемый файл и запускаю его.(рис. [??]).



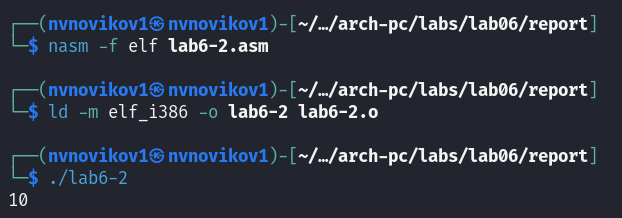
Запуск новой программы

Аналогично предыдущему примеру изменяю символы на числа.(рис. [??]).



Изменение программы

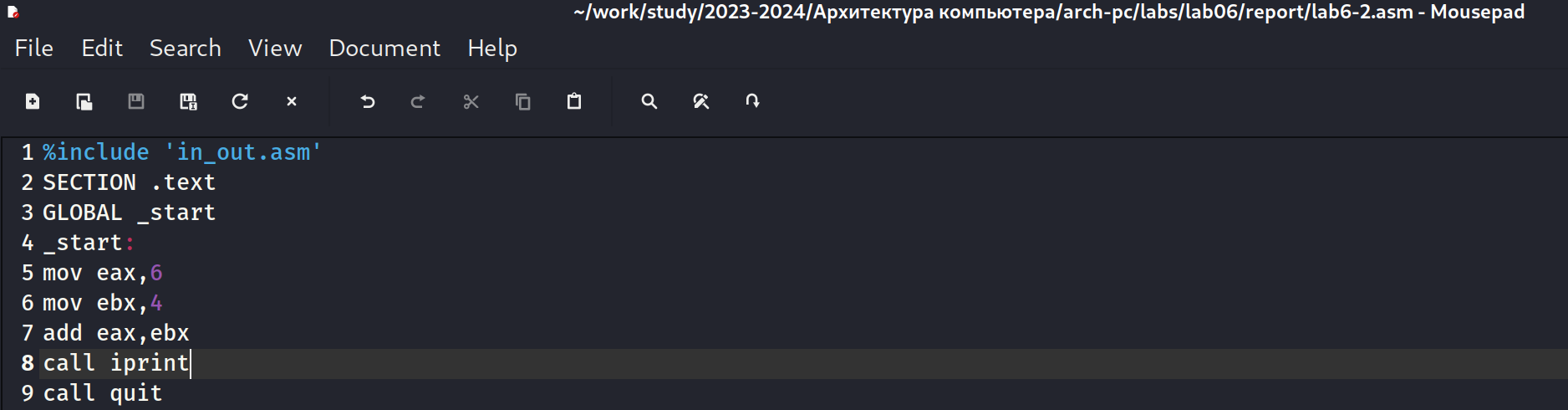
Создаю исполняемый файл и запускаю его.(рис. [??]).



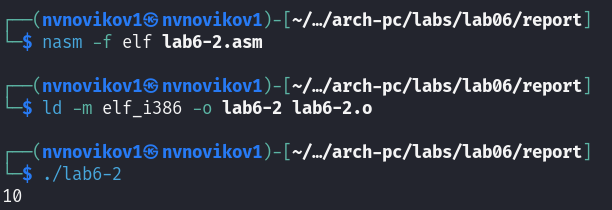
Запуск исправленной программы

При исполнении программы было получено число 10.

Заменяю функцию iprintLF на iprint. Создаю исполняемый файл и запускаю его.(рис. [??]).(рис. [??]).



Изменение файла

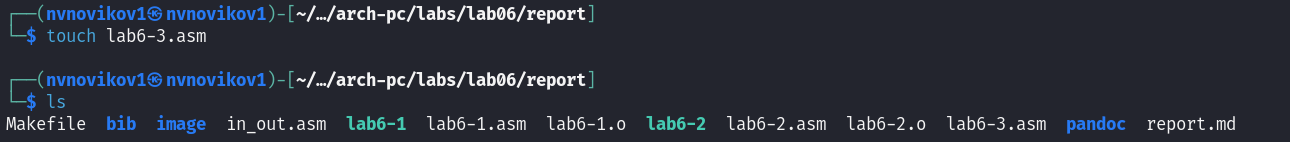


Запуск программы

Вывод функции iprint отличается от iprintLF тем, что выведенное сообщение не переносится на слудующую строку.

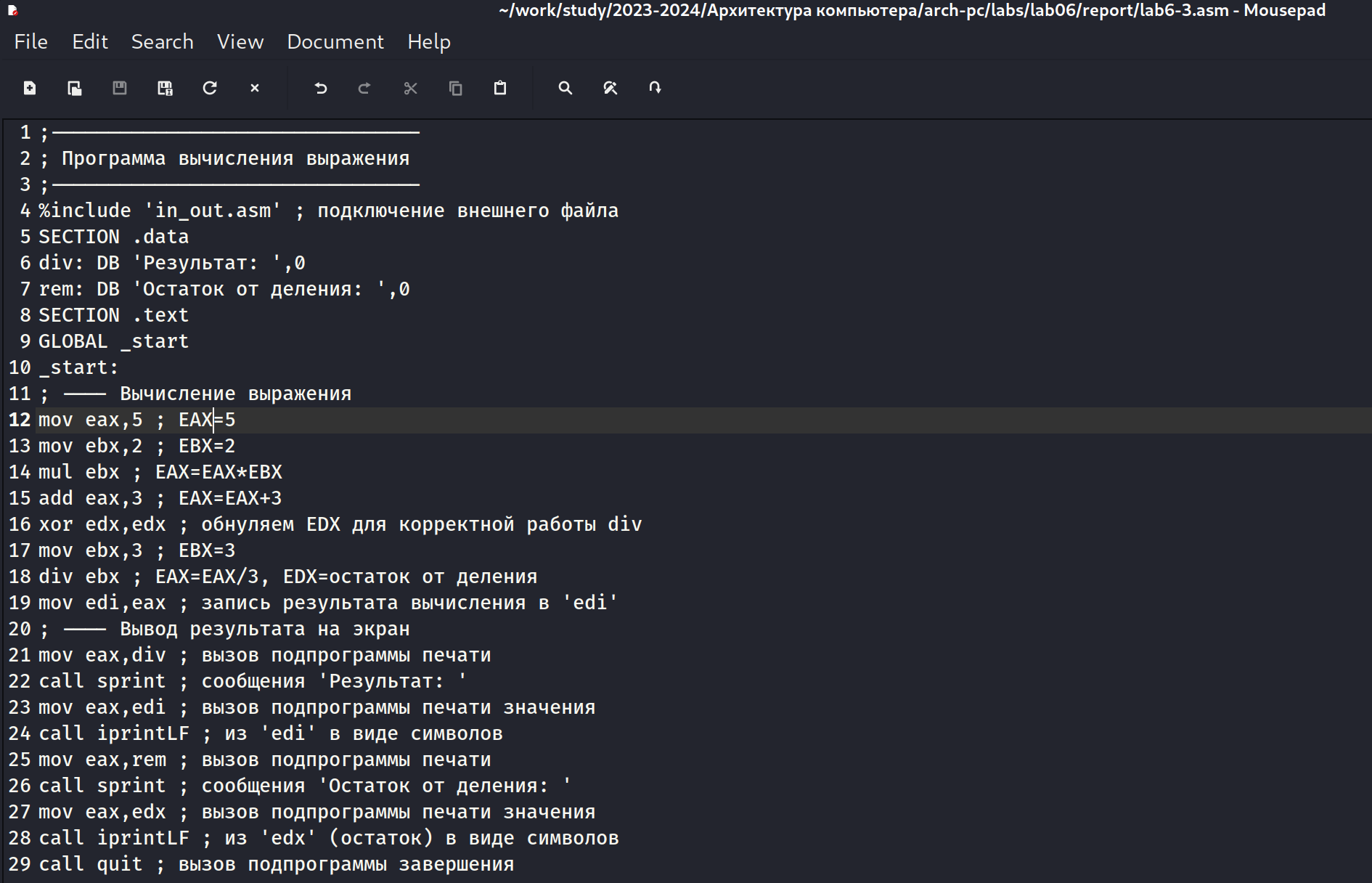
## 4.2 Выполнение фрифметических операций в NASM

Создаю файл lab6-3.asm в каталоге ~/work/arch-pc/lab06.(рис. [??]).



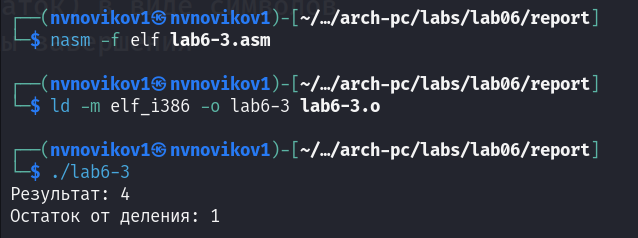
Создание файла

После внимательного прочтения текста программы из листинга 6.3 ввожу его в lab6-3.asm.(рис. [??]).



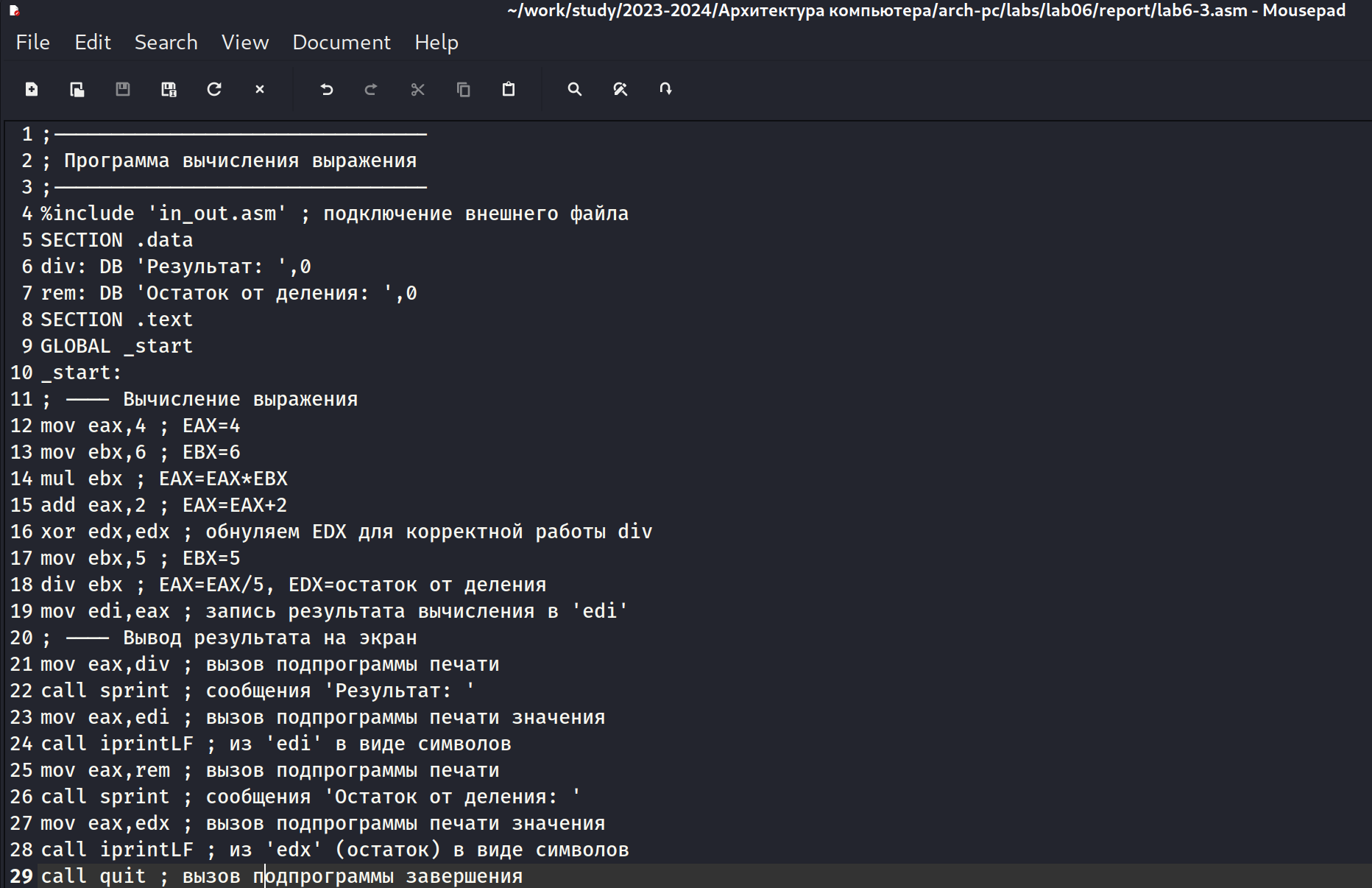
Ввод программы из листинга 6.3

Создаю исполняемый файл и запускаю его.(рис. [??]).



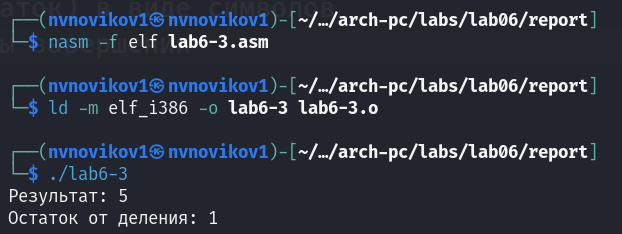
Запуск программы

Изменяю тест программы для вычисления выражения f(x)=(4\*6+2)/5.(рис. [??]).



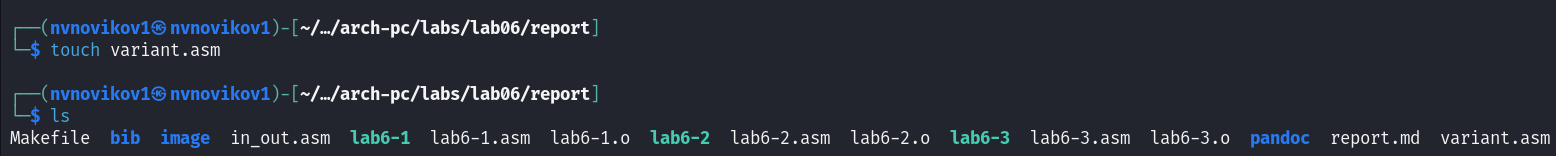
Редактирование текста программы

Создаю исполняемый файл и запускаю его.(рис. [??]).



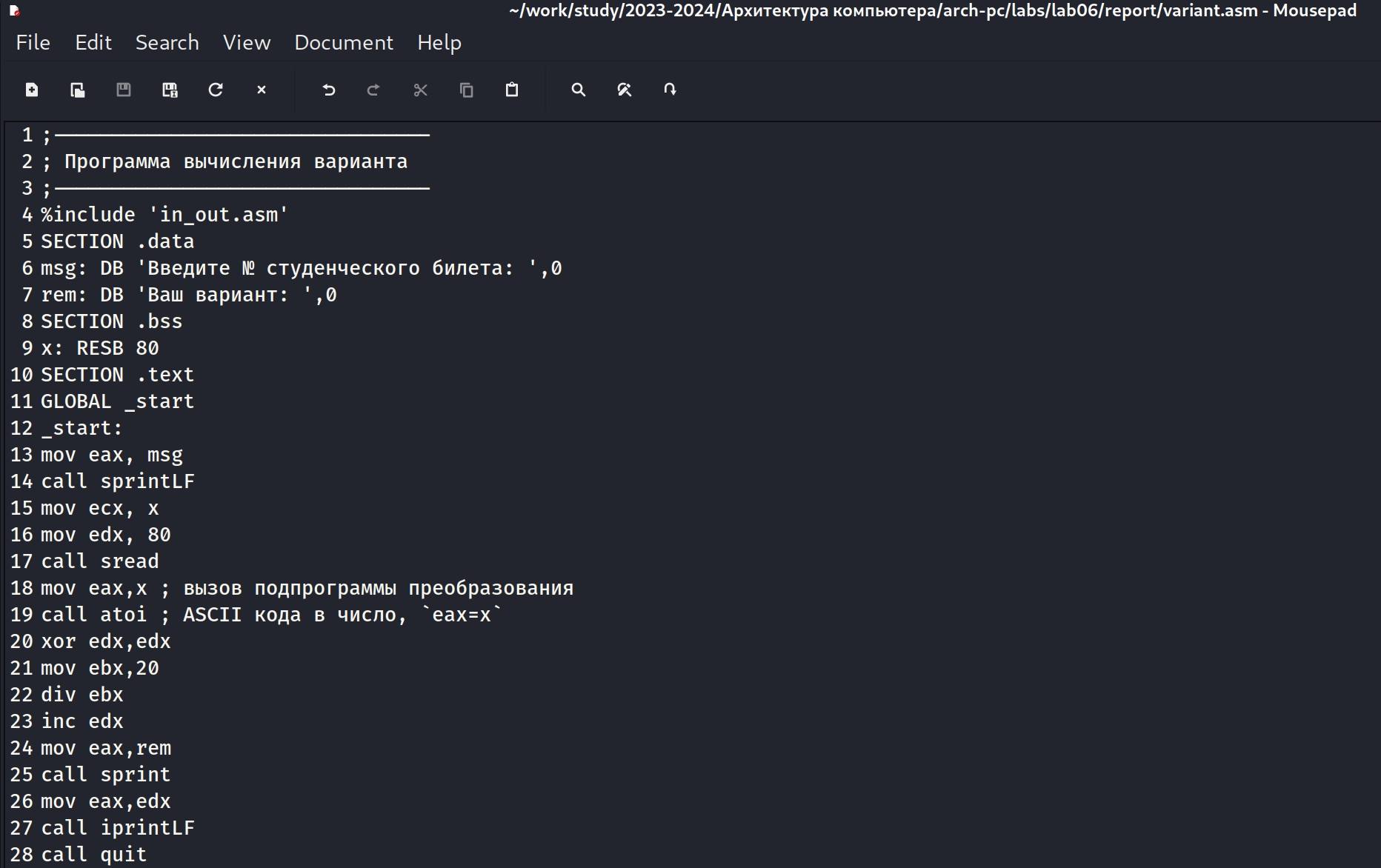
Запуск программы

Создаю файл variant.asm в каталоге ~/work/arch-pc/lab06.(рис. [??]).



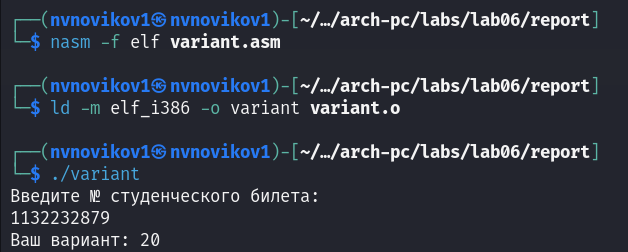
Создание файла variant.asm

Внимательно изучаю текст программы из листинга 6.4 и ввожу в файл variant.asm.(рис. [??]).



Создание файла variant.asm

Создаю исполняемый файл и запускаю его.(рис. [??]).



Запуск файла

Ответы на вопросы

1. За вывод сообщение “Ваш вариант” отвечают строки: mov eax,rem call sprint
2. Эти строки используются чтобы считать x.
3. Call atoi преобразовывает код ASCII в целое число
4. За вычисление варианта отчечают строки: xor edx,edx mov ebx,20 div ebx inc edx
5. Остаток от деление записывается в регистр edx.
6. Инструкция inc edx используется для того, чтобы увеличить значение регистра edx на 1.
7. Для вывода на экран результата выислений используются строки: mov eax,edx call iprintLF

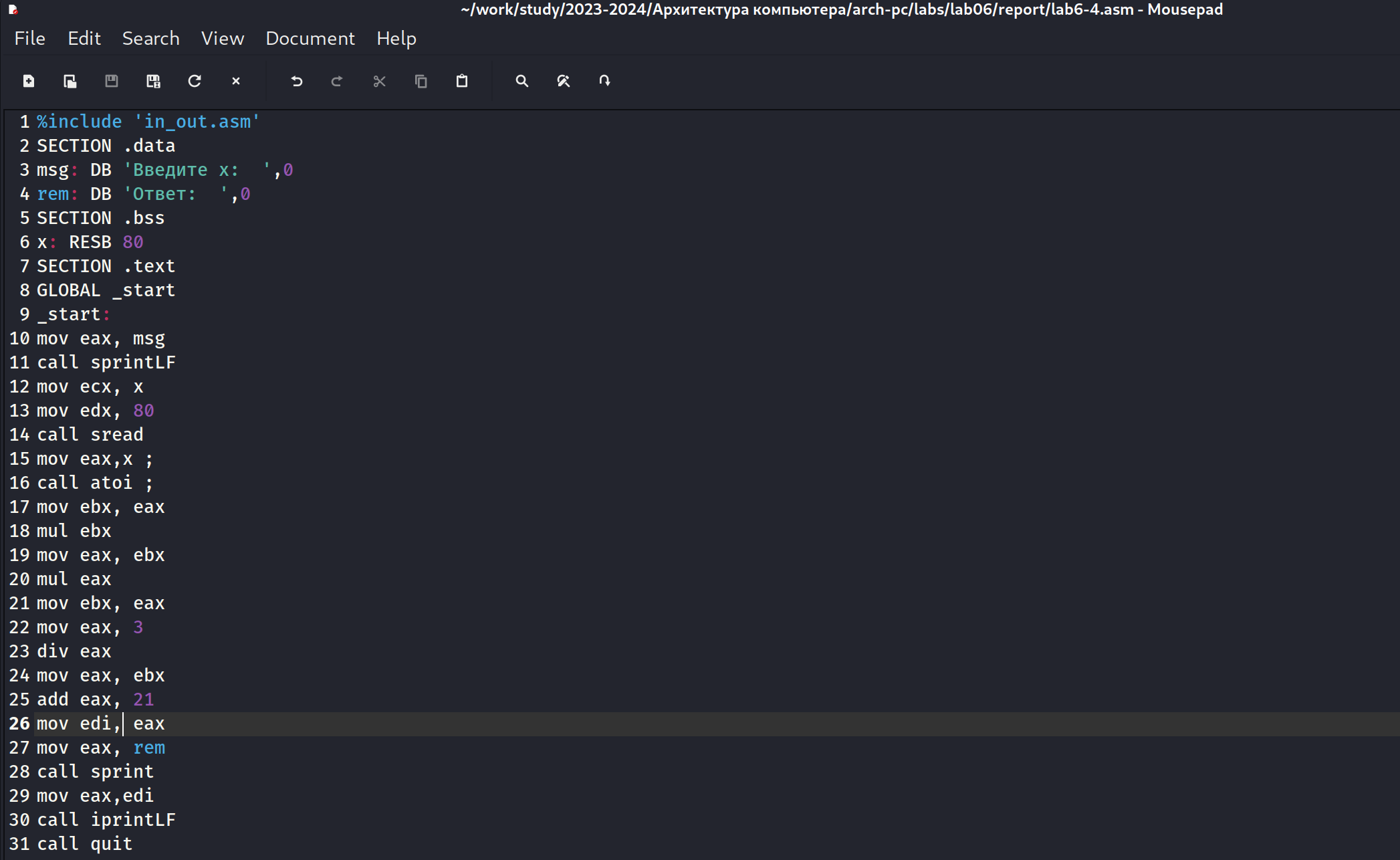
#Задание для самостоятельной работы

Создаю файл lab6-4.asm. Проверяю создание файла.(рис. [??]).



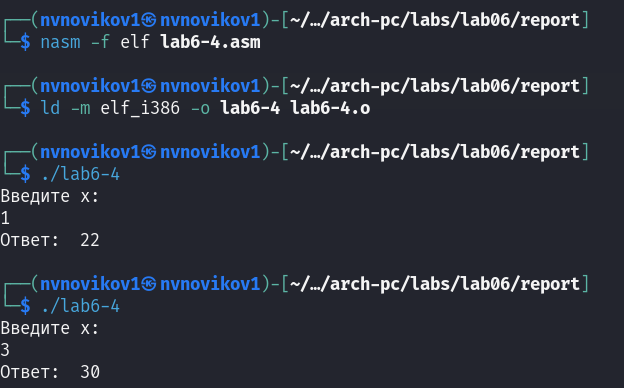
Создание файла

Открываю созданную файл и начинаю печатать в него текст программы для вычисления (x^3)\*(1/3)+21 (вариант 20)(рис. [??]).



Ввод программы

Далее сохранию файл, создаю исполняемый файл и запускаю его. (рис. [??]).



Проверка для значения x1 и x2

Всё верно работает.

# 5 Выводы

После выполнения данной работы я освоил фрифметические инструкции языка ассемблера NASM

# Список литературы