Отчет по лабораторной работе #6

Простейший вариант

Баженов Тимур

Содержание

Список иллюстраций

Список таблиц

# Цель работы

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

# Задание

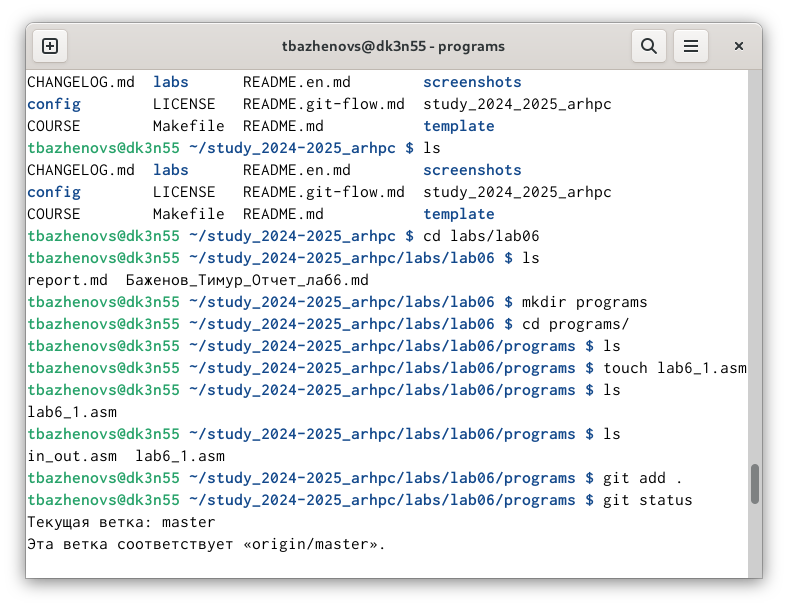
1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

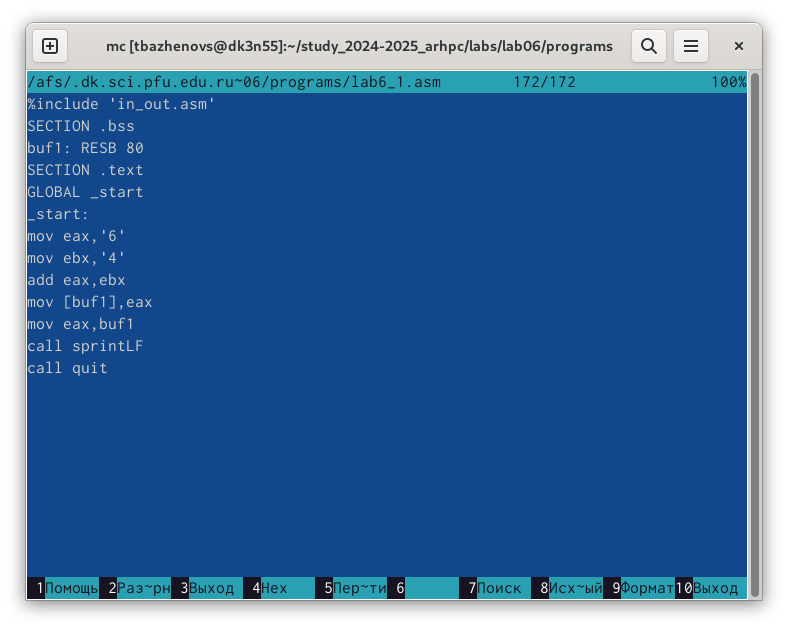
# Теоретическое введение

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.

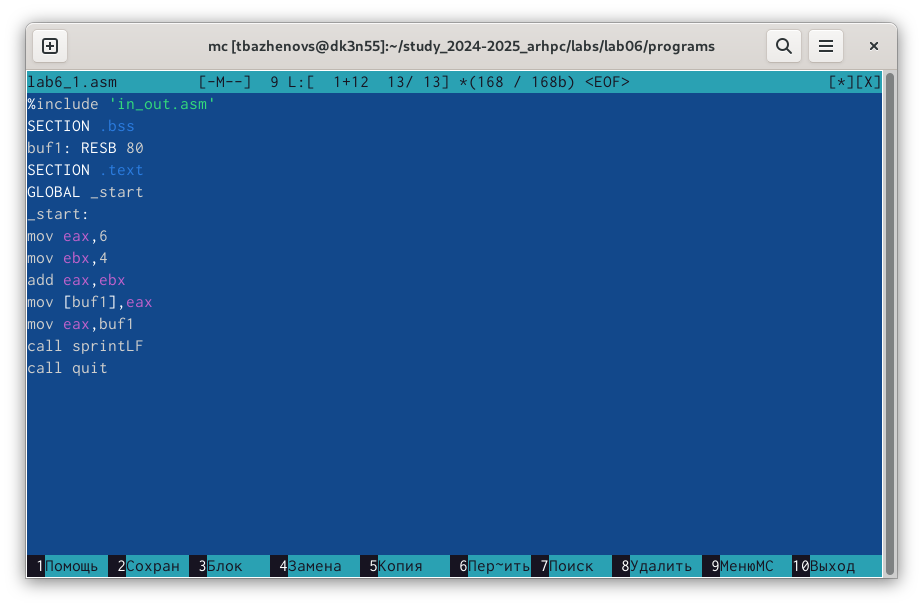
Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно 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.

# Выполнение лабораторной работы

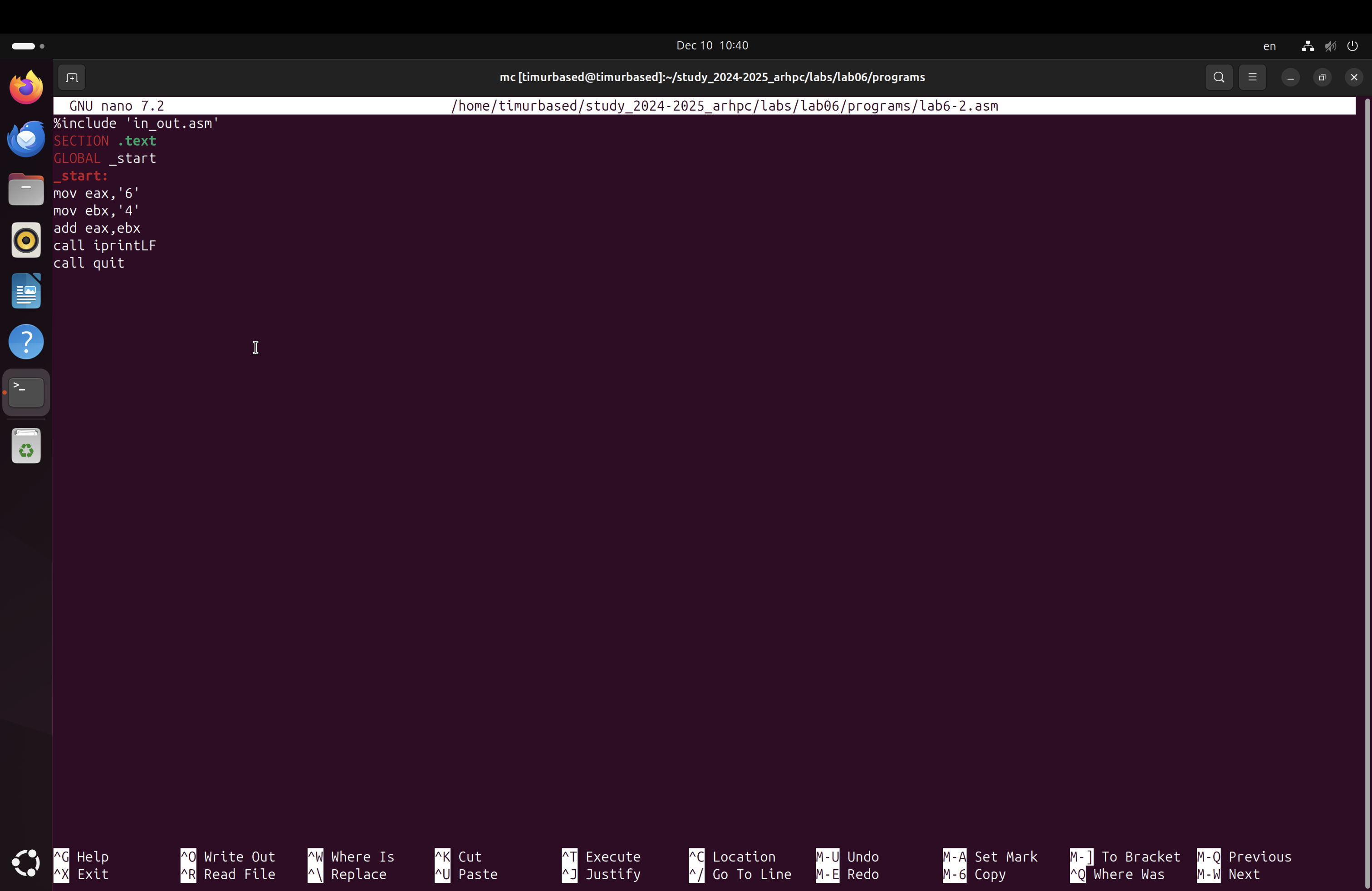
Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. **¿fig:001?**). Создаю каталог и файл lab6\_1.asm 

Открываем его в режиме редактирования и вставляем туда код. 

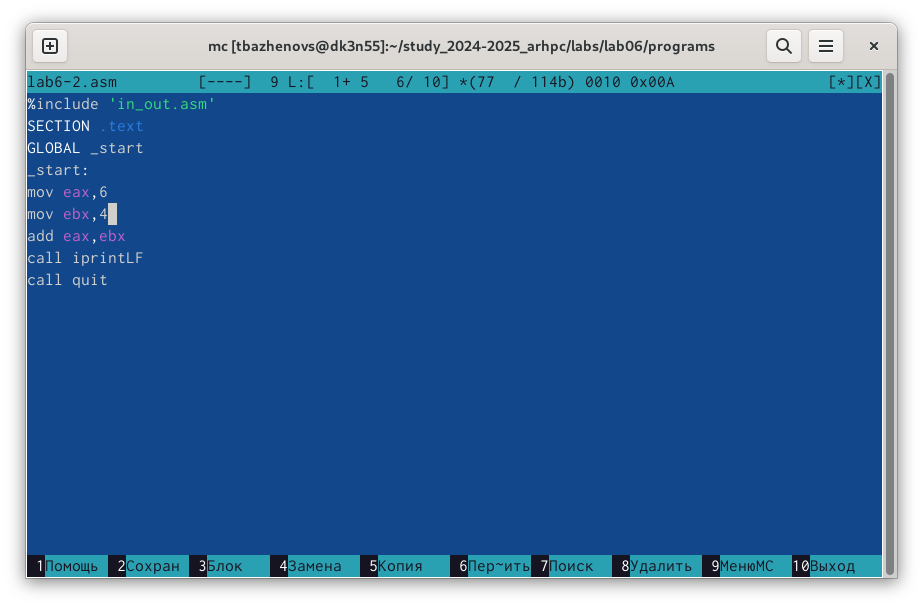
Запускаем файл. Видим символ j, потому что программа вывела то, что соответствует по системе ASCII сумме двоичных кодов символов 4 и 6 

Изменяю в программе символы “6” и “4” на 6 и 4. 

Выполняем запуск программы и замечаем, что вывелся символ с кодом 10, это символ перевода строки, а данный символ не отображается при выводе на экран. 

Создаю файл lab6-2.asm и вставляю в него код из ТУИС. 

Запускаем файл, и получаем результат 106. 

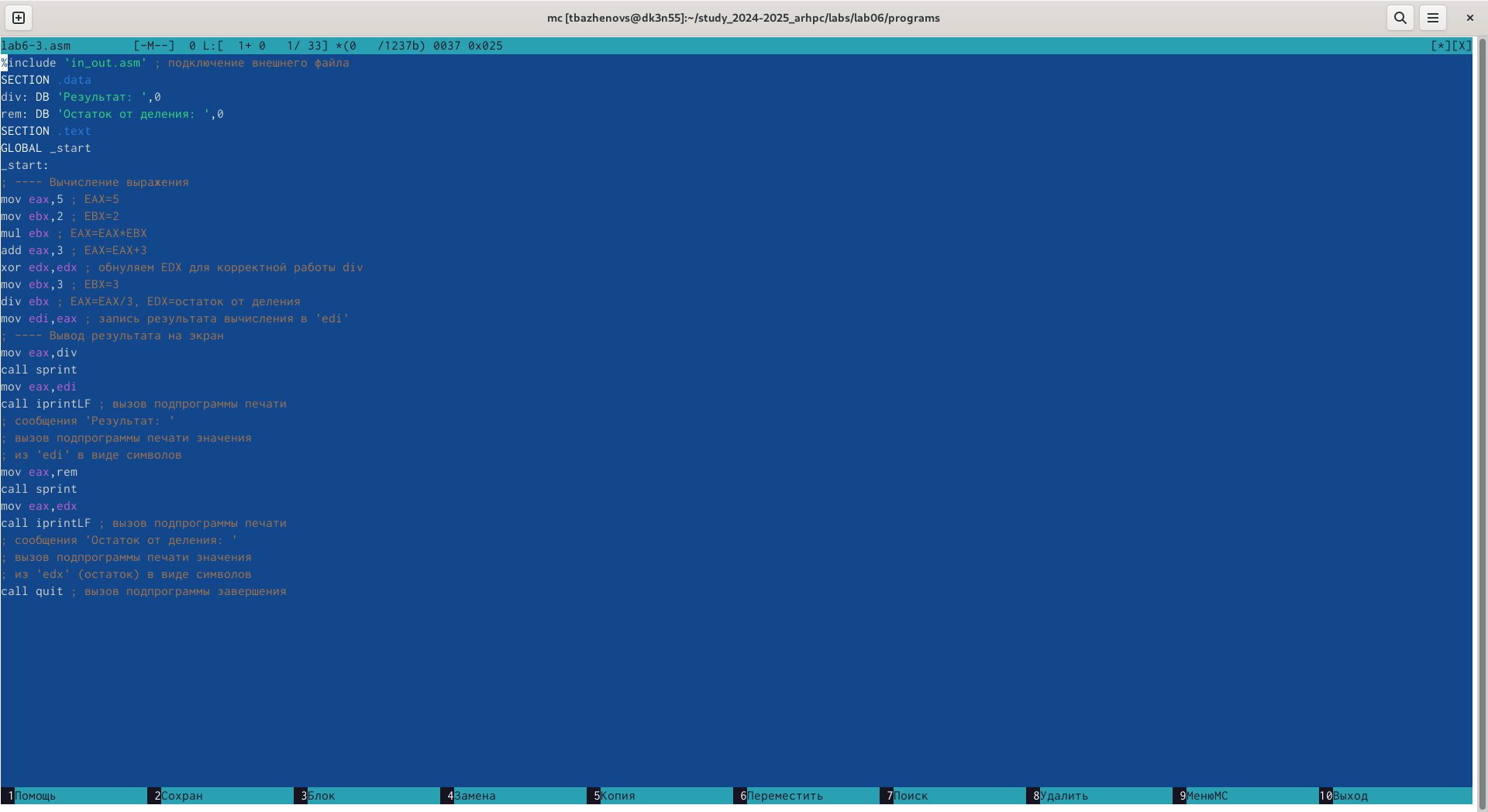
Изменяю в программе символы “6” и “4” на 6 и 4. 

Запускаем файл, и получаем результат 10, это сумма 6 и 4. 

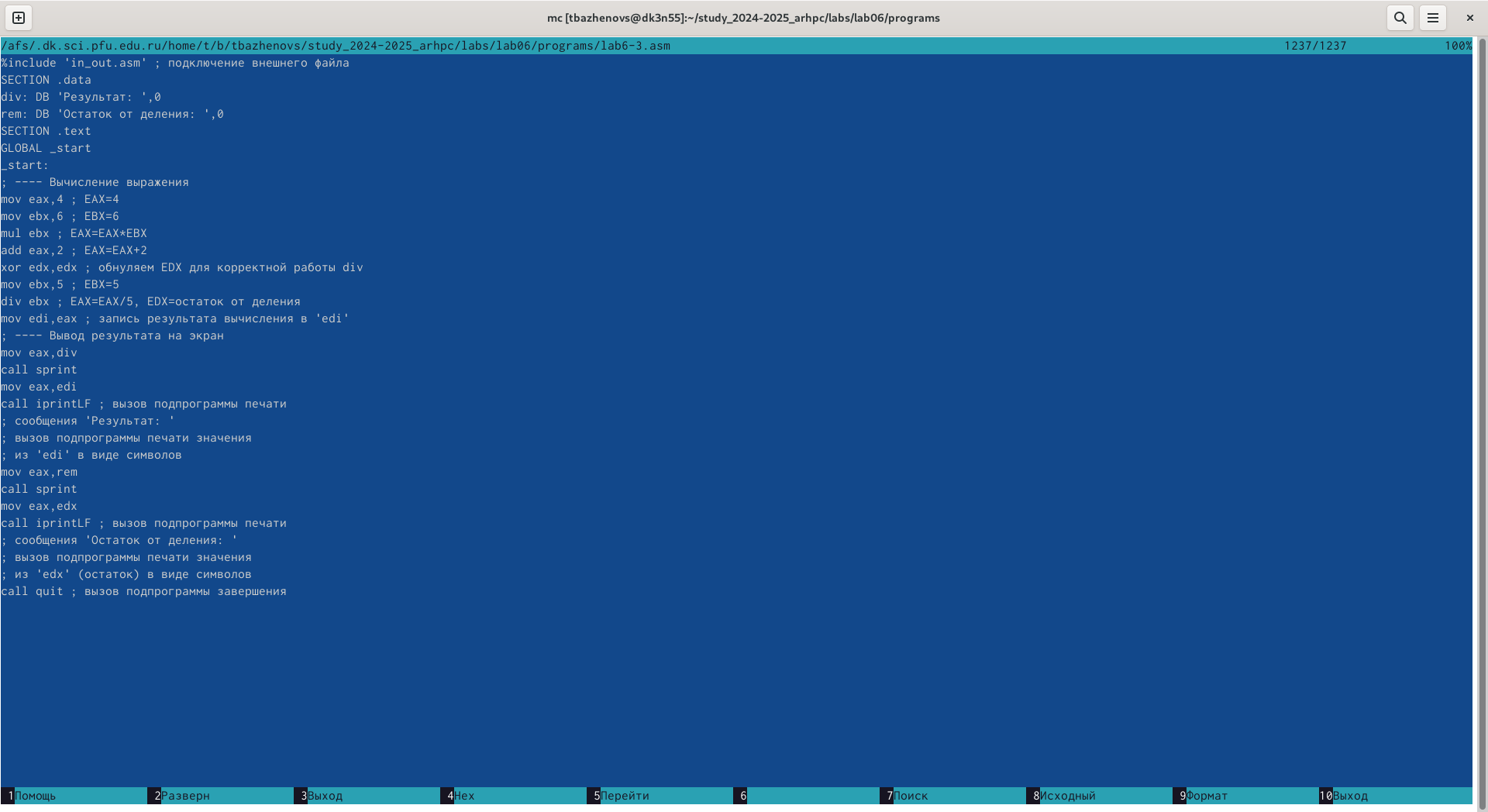
Если в выводе поменять функции printLF на iprint, то текст не выведится с новой строки.

# Выполнение арифметических операций в NASM

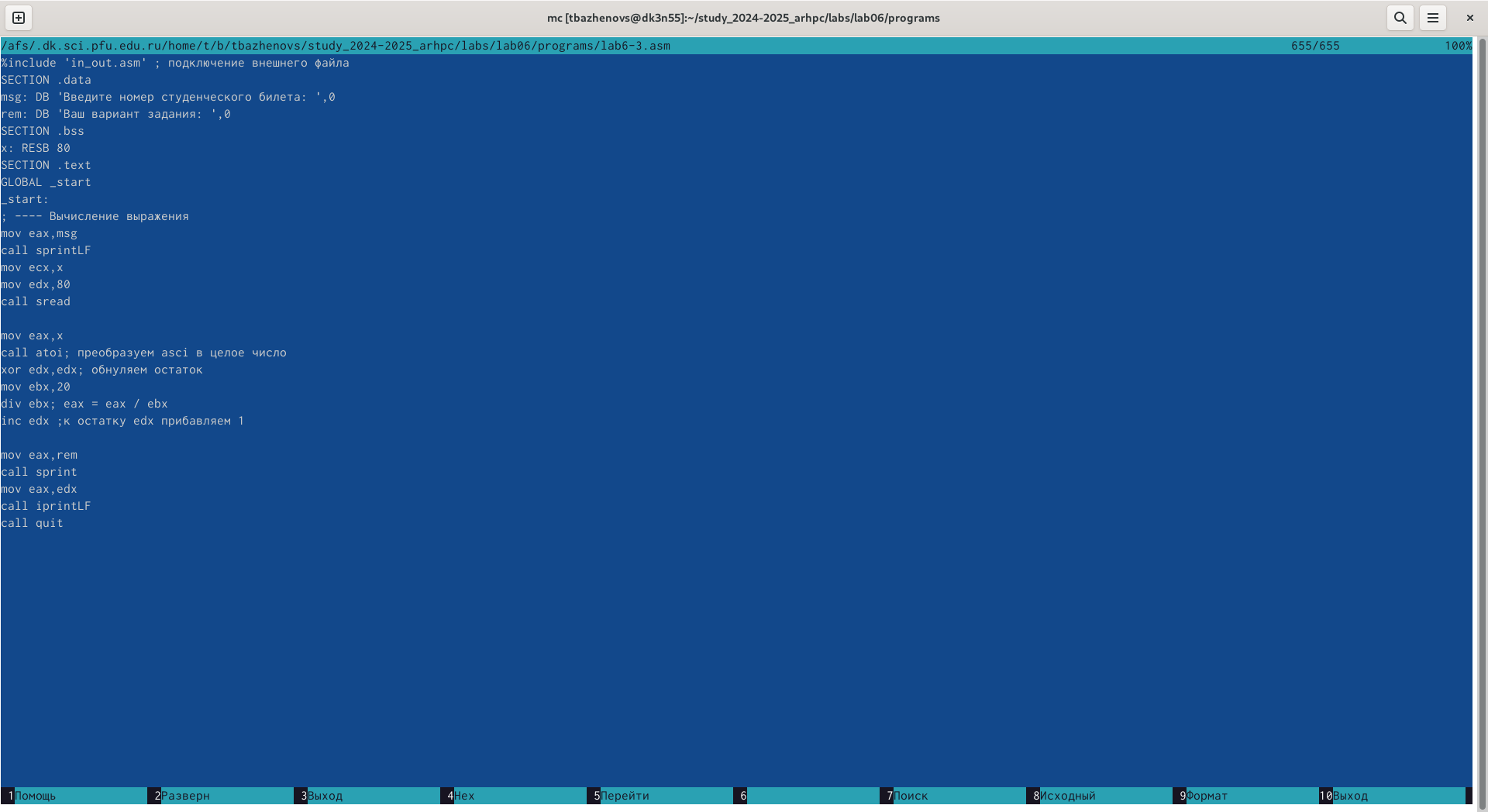
Создаем файл lab6-3.asm. 

Вставляем туда код и ТУИС. 

Компонуем и запускаем. Получаем 4, и остаток 1. 

Изменяем текст программы для вычисления выражения f = (4\*6+2)/5 

Компонуем и проверяем. 

Создаем файл variant и вставляем туда код. 

Запускаем и проверяем результат. 

# Ответы на вопросы

1. За вывод сообщения “Ваш вариант” отвечает строка:

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. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax.
3. За вычислени варианта, отвечают строки

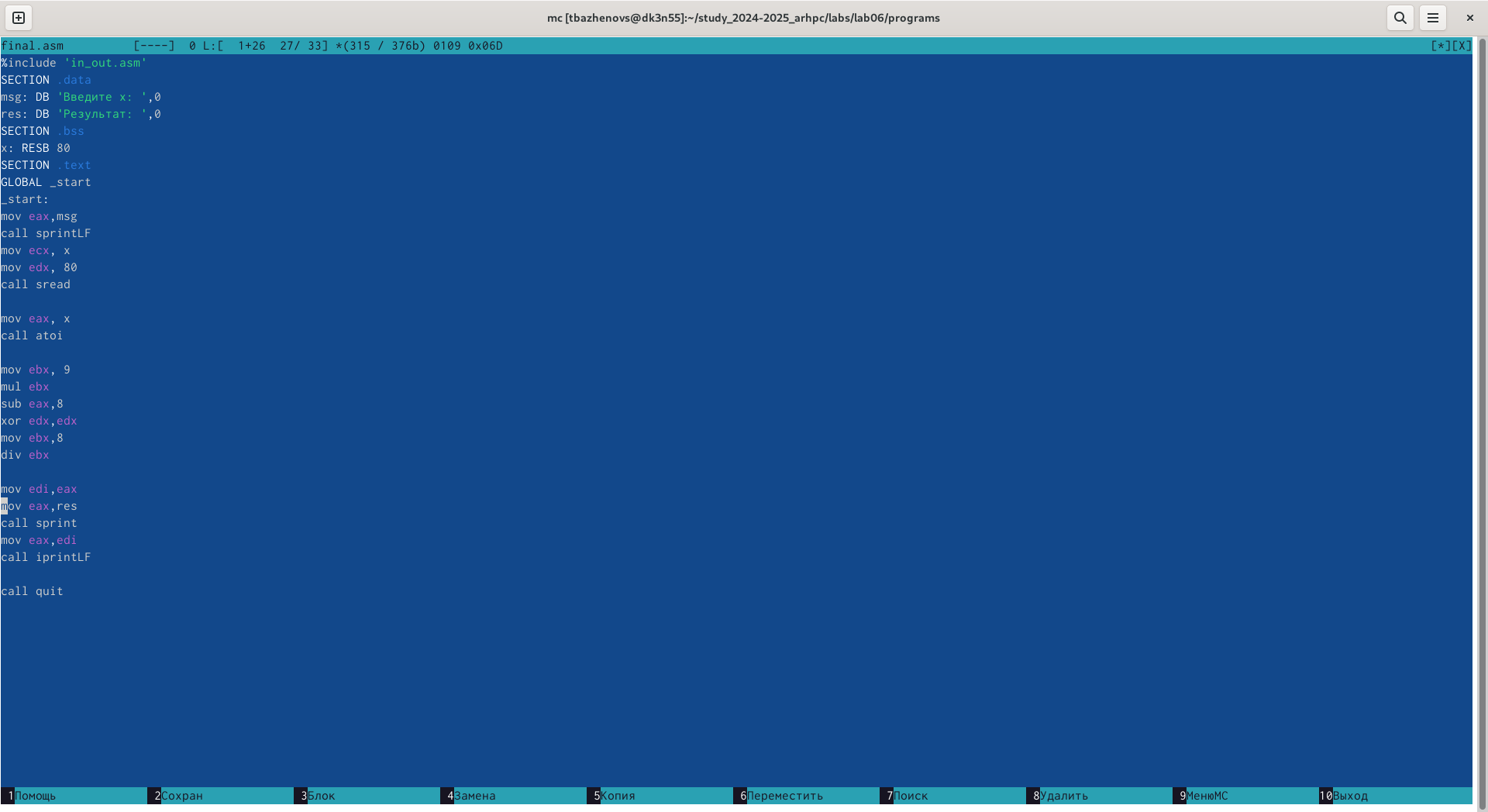
mov ebx,20  
div ebx; eax = eax / ebx  
inc edx ;к остатку edx прибавляем 1

1. В регистр EDX.
2. Инструкция INC отвечает за инкремирование значения переменных на 1.
3. За вывод на экран, отвечают строки:

mov eax,edx  
call iprintLF

# Задание для самостоятельной работы

Создаем файл final.asm. 

Далее пишу код для выполнения индивидуального задания. 

Выполняем запуск программы 

# Выводы

При выполнении данной работы мы познакомились с арифметическими операциями в языке программирования Assembly NASM.

# Список литературы

https://esystem.rudn.ru/pluginfile.php/2089086/mod\_resource/content/0/Лабораторная%20работа%20№6.%20Арифметические%20операции%20в%20NASM..pdf