Отчёт по лабораторной работе № 6

Дисциплина: Архитектура компьютера

Кижваткина Анна Юрьевна

Содержание

# 1 Цель работы

Целью данной лабораторной работы является освоение арифметических инструкций языка ассемблера NASM.

# 2 Задание

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

# 3 Теоретическое введение

Большинство указаний на языке ассемблера требуют обработки операндов. Адрес операнда указывает на место, где хранятся данные, которые необходимо обработать. Эти данные могут быть хранящимися в регистре или ячейке памяти.

Адресация по регистрам – данные хранятся в регистрах, и в команде используются имена этих регистров, например: mov ax, bx.

Непосредственная адресация – значение операнда указывается непосредственно в команде, например: mov ax, 2.

Адресация памяти – операнд указывает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой нужно выполнить операцию.

Ввод информации с клавиатуры и вывод ее на экран происходит в символьном виде в соответствии с таблицей символов ASCII.

Согласно стандарту ASCII, каждый символ кодируется одним байтом. Среди инструкций NASM отсутствует инструкция для вывода чисел (не в символьном виде). Поэтому для вывода чисел нужно сначала преобразовать цифры в ASCII-коды и выводить на экран эти коды, а не сами числа.

Подобная ситуация возникает и при вводе данных с клавиатуры – введенные данные будут представлять символы, что затрудняет арифметические операции с ними. Для решения этой проблемы необходимо преобразовать ASCII символы в числа и наоборот.

# 4 Выполнение лабораторной работы

1. Символьные и численные данные в NASM.

Создаем каталог для программ лабораторной работы №6 (рис. 1).

Рис. 1: Создание каталога

Рис. 1: Создание каталога

Переходим в созданный каталог. (рис. 2).

Рис. 2: Перемещение в каталог

Рис. 2: Перемещение в каталог

Создаем файл lab6-1.asm. (рис. 3).

Рис. 3: Создание файла

Рис. 3: Создание файла

Рассмотрим примеры программ вывода символьных и численных значений. Вводим в файле программу с листинга 6.1 (рис. 4).

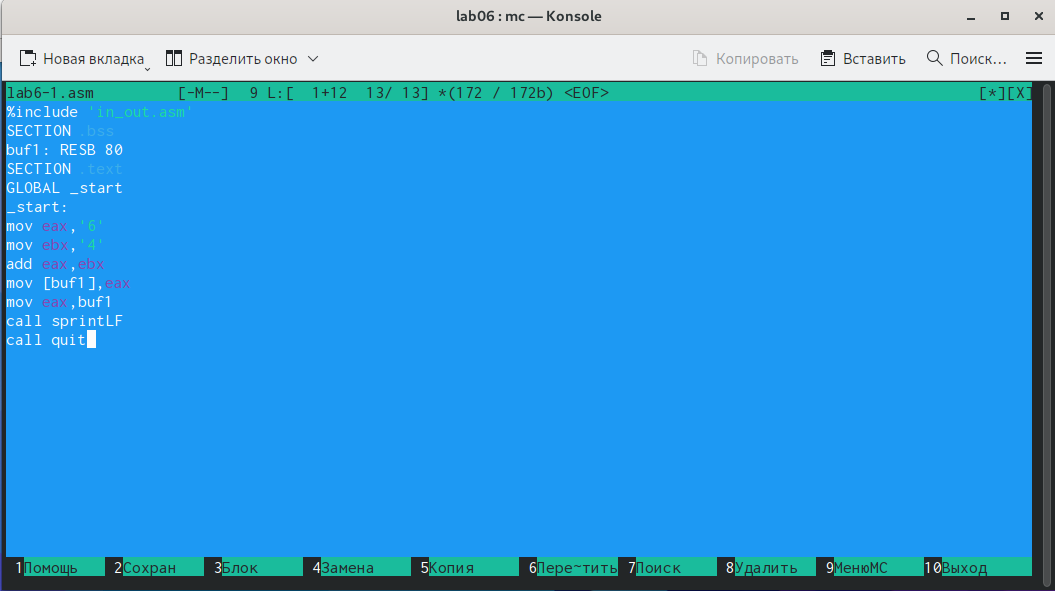


Рис. 4: Изменение файла

Скачиваем и переносим файл in\_out.asm в каталог лабораторной работы 6. (рис. 5).

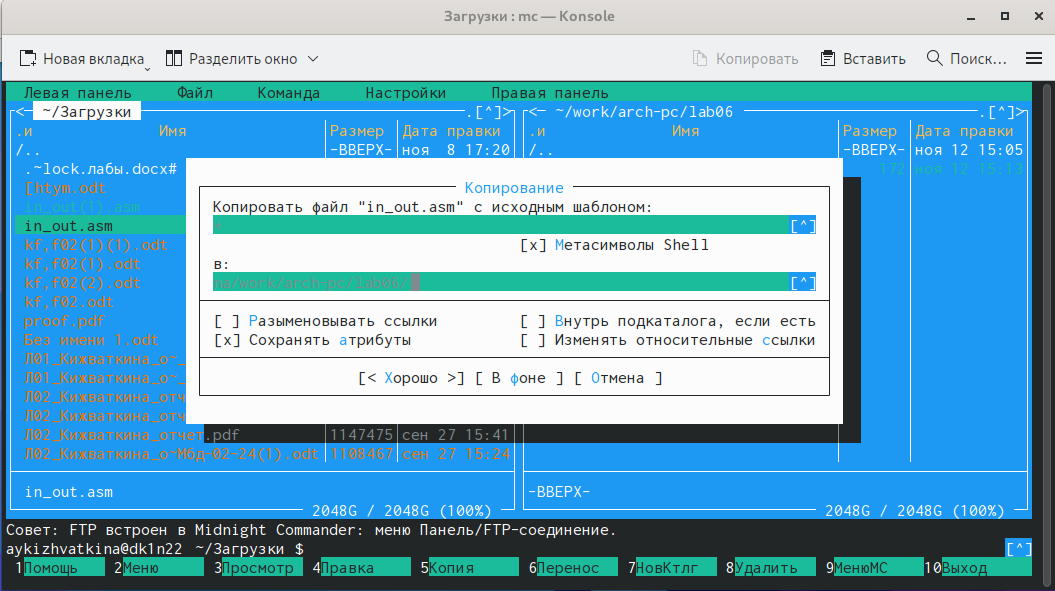


Рис. 5: Перенос файла

Создаем исполняемый файл и запускаем его. Программа выводи символ j так как программа выводит символ, соответствующий сумме двоичных кодов 6 и 4. (рис. 6).

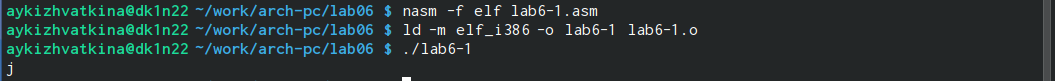


Рис. 6: Создание исполняемого файла и его запуск

Изменяем в тексте программы символы ‘6’ и ’4’ на 6 и 4. (рис. 7).

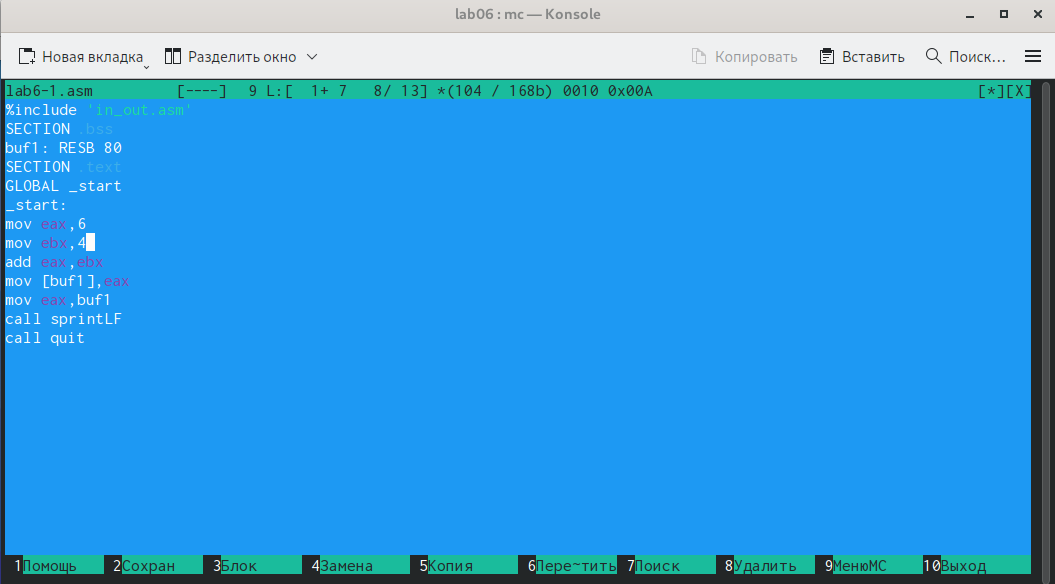


Рис. 7: Изменение программы

Создаем исполняемый файл и запускаем его. Теперь вывелся символ с кодом 10. Это символ перевода строки, он не отображается на экране. (рис. 8).

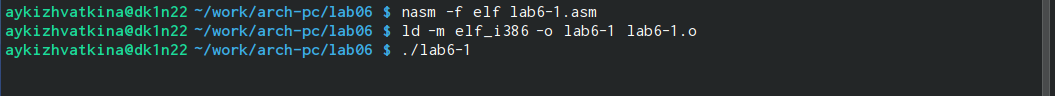


Рис. 8: Создание исполняемого файла и его запуск

Создаем файл lab6-2.asm. (рис. 9).

Рис. 9: Создание файла

Рис. 9: Создание файла

Вводим в файл текст программы из листинга 6.2. (рис. 10).

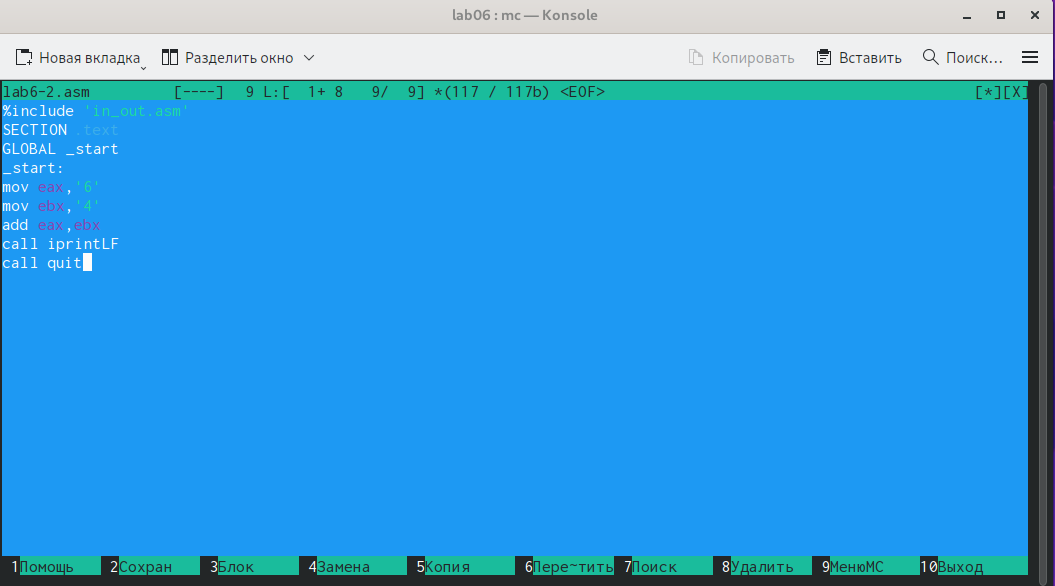


Рис. 10: Ввод программы

Создаем исполняемый файл и запускаем его. В результате работы программы мы получим число 106 т.к команда add складывает коды символов, но функция iprintLF позволяет вывести число, а не символ, кодом которого является это число. (рис. 11).

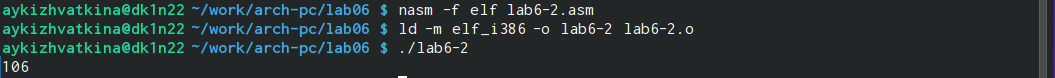


Рис. 11: Создание исполняемого файла и его запуск

Изменяем в тексте программы символы ‘6’ и ’4’ на 6 и 4. (рис. 12).

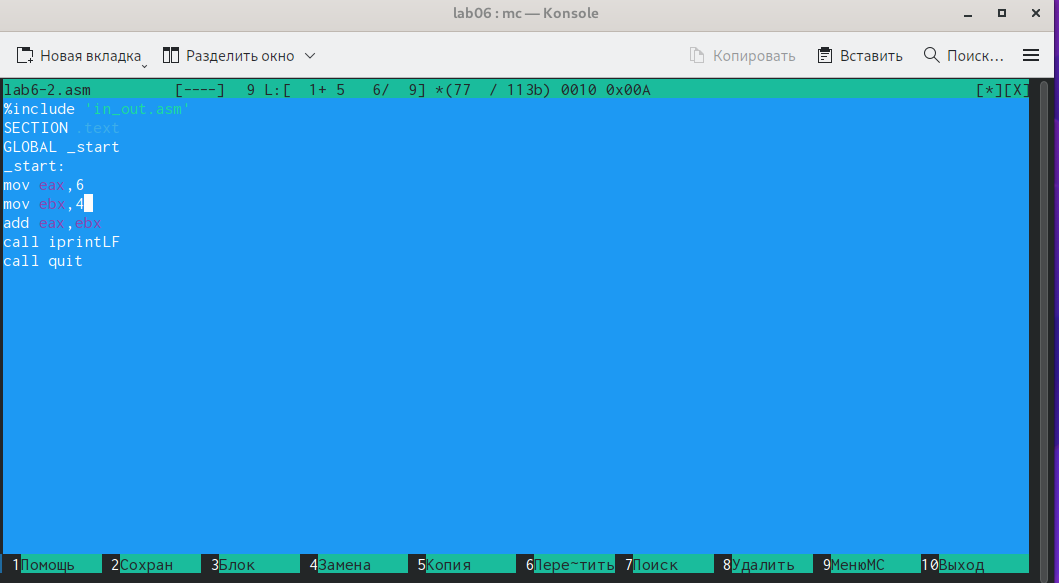


Рис. 12: Изменение программы

Создаем исполняемый файл и запускаем его. Теперь программа складывает не коды, а сами числа, поэтому результатом является число 10. (рис. 13).

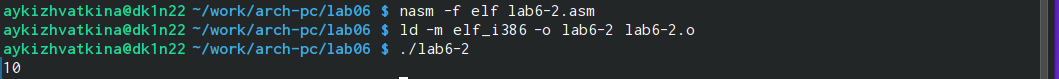


Рис. 13: Создание исполняемого файла и его запуск

Заменяем в тексте программы функцию iprintLF на iprint. (рис. 14).

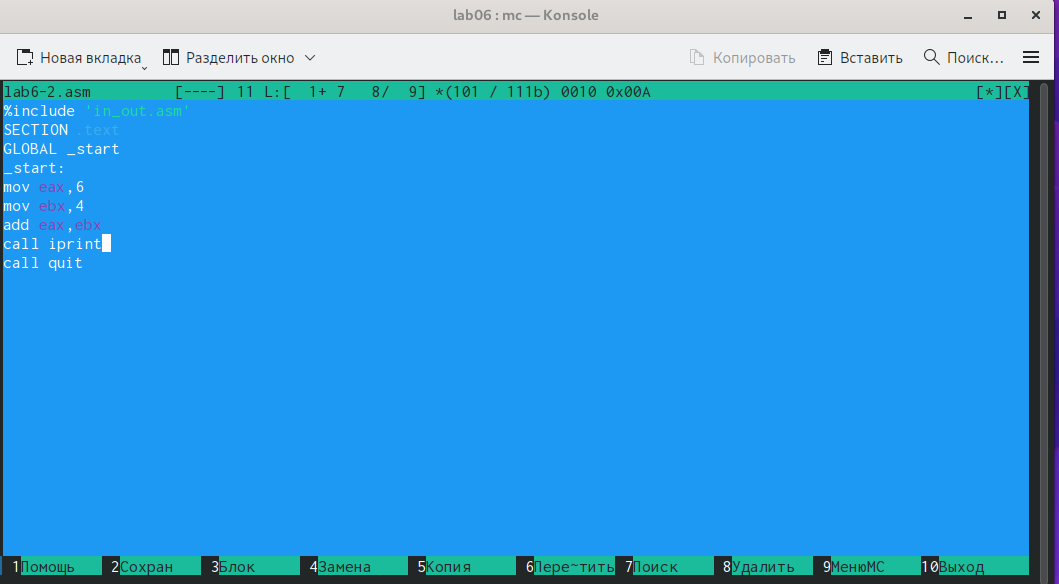


Рис. 14: Изменение программы

Создаем исполняемый файл и запускаем его. Вывод не изменился, но символ переноса строки не отображается. (рис. 15).

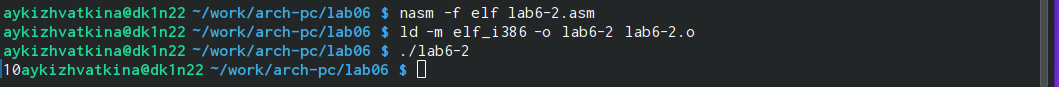


Рис. 15: Создание исполняемого файла и его запуск

1. Выполнение арифметических операций в NASM.

Создаем файл lab6-3.asm. (рис. 16).

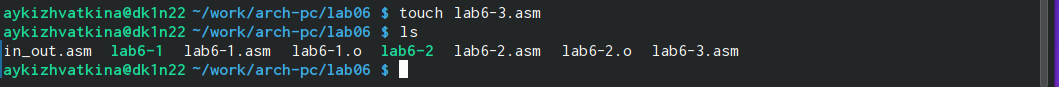


Рис. 16: Создание файла

Вводим в созданный файл текст программы из листинга 6.3. (рис. 17).

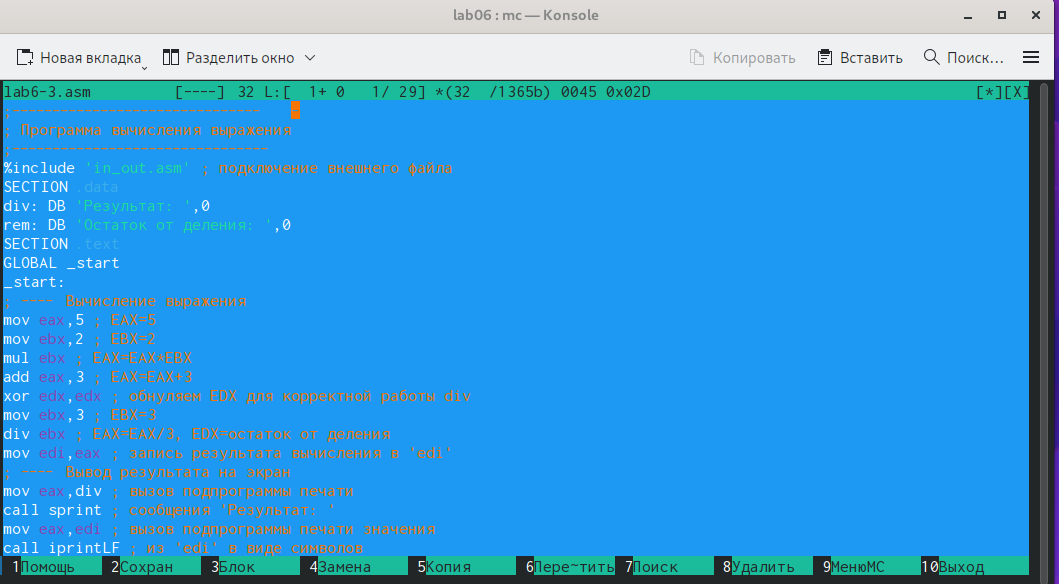


Рис. 17: Ввод программы

Создаем исполняемый файл и запускаем его. (рис. 18).

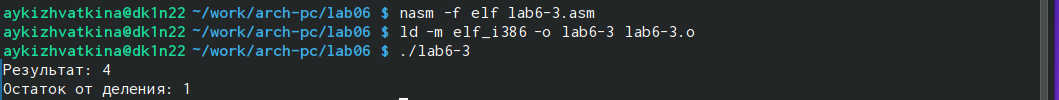


Рис. 18: Создание исполняемого файла и его запуск

Изменяем текст программы для вычисления выражения. (рис. 19).

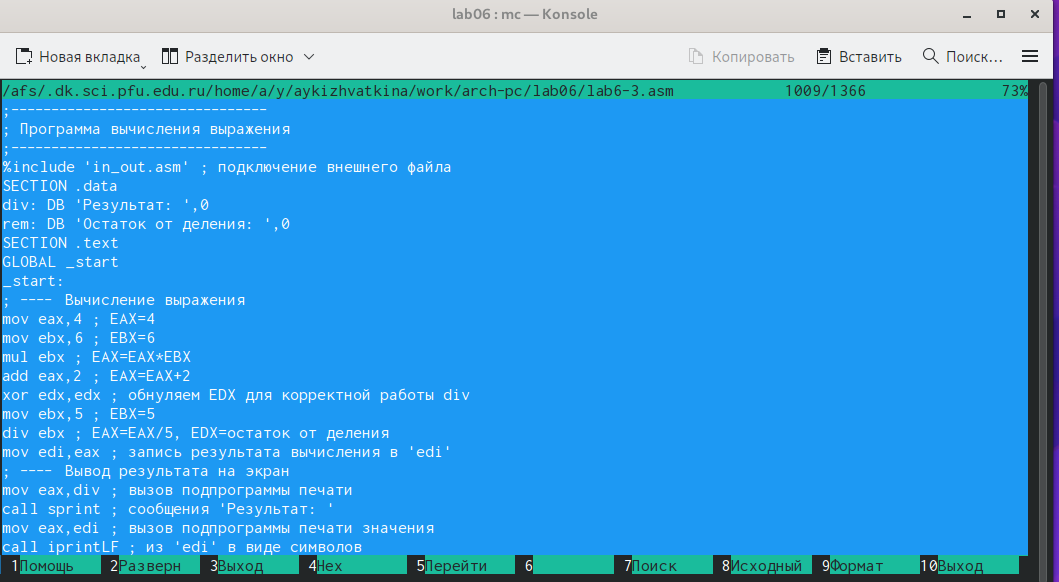


Рис. 19: Изменение программы

Создаем исполняемый файл и проверяем его работу. (рис. 20).

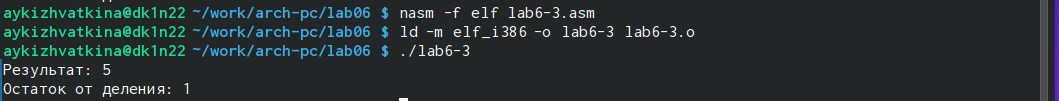


Рис. 20: Создание исполняемого файла и его запуск

Создаем файл variant.asm с помощью touch. (рис. 21).

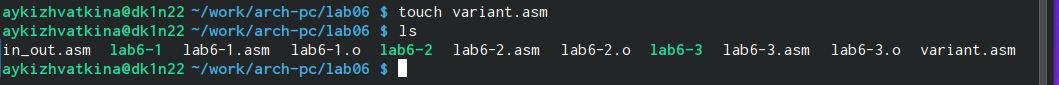


Рис. 21: Создание файла

Вводим в файл текст программы из листинга 6.4. (рис. 22).

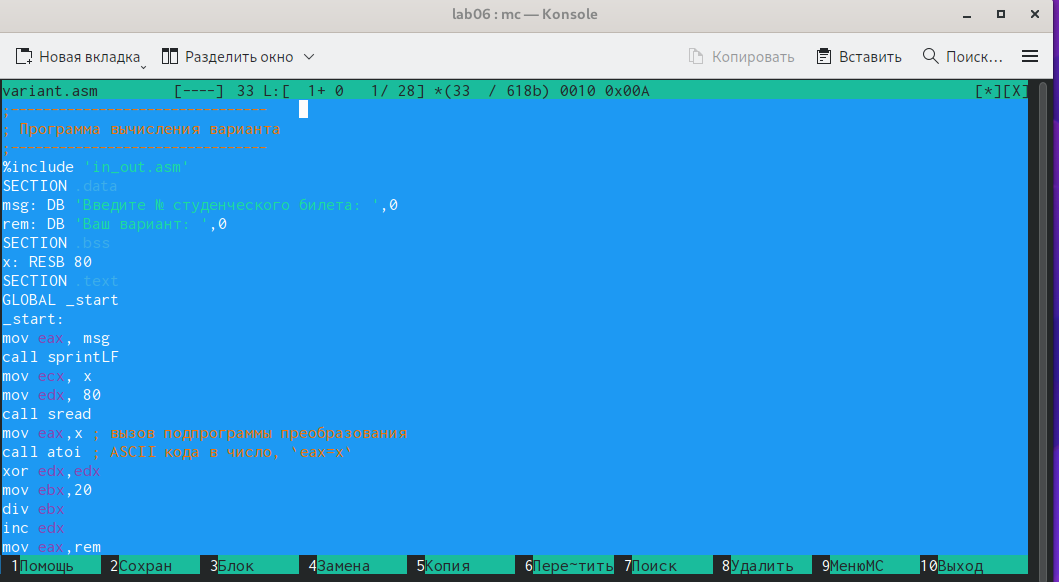


Рис. 22: Ввод программы

Создаем и запускаем исполняемый файл. Вводим номер своего студенческого билета проверяем выведенный ответ аналитически. (рис. 23).

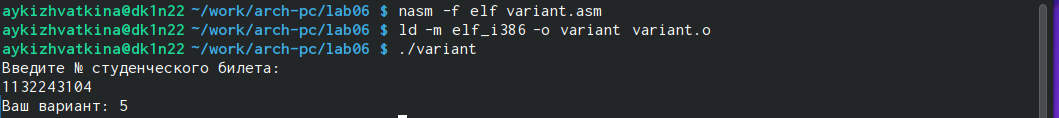


Рис. 23: Создание исполняемого файла и его запуск

Ответы на вопросы:

1. За вывод сообщения «Ваш вариант» отвечает данная 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 регистр edx длины вводимой строки call spread – вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры.
2. Call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует asci-код символа в целое число и записывает результат в регистр eax.
3. За вычисление варианта отвечают данные строки:

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

1. При выполнении инструкции div ebx остаток от деления записывается в регистр edx.
2. Инструкция inc edx увеличивает значение регистра edx на 1.
3. За вывод результата на экран отвечают данные строки:

mov eax,edx  
call iprintLF

1. Выполнение самостоятельной работы.

Создаем файл lab6-4 с помощью touch. (рис. 24).

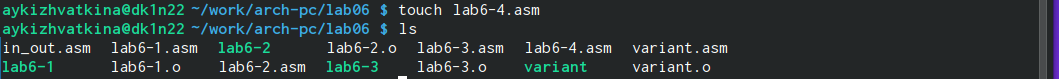


Рис. 24: Создание файла

Вводим в файл текст программы для вычисления выражения под номером 18. (рис. 25).

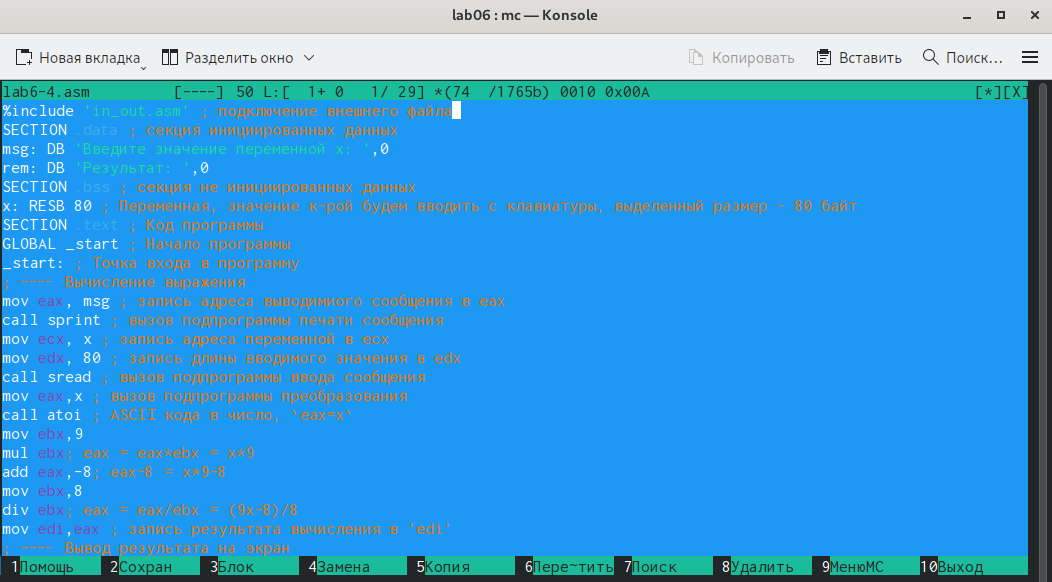


Рис. 25: Ввод программы

Создаем и запускаем файл. Проверяем оба значения. (рис. 26).

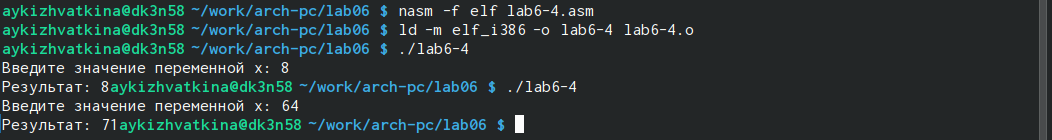


Рис. 26: Создание исполняемого файла и его запуск

# 5 Выводы

С помощью данной лабораторной работы мы освоили арифметические инструкции языка ассемблера NASM.