ОТЧЕТ по лабораторной работе № 6

дисциплина: Архитектура компьютера

Студент: Лисенков Е.Р.

Содержание

# 1 Цель работы

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

# 2 Задания

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

# 3 Теоретическое введение

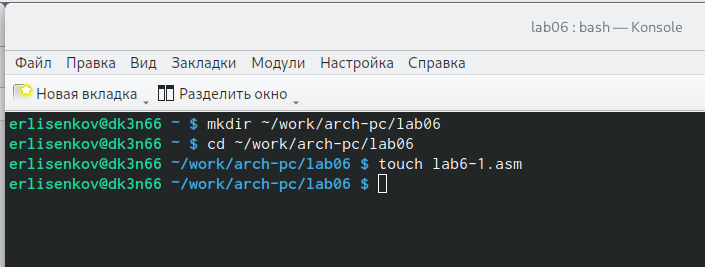
Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. - Регистровая адресация – операнды хранятся в регистрах и в команде используются имена 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. 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

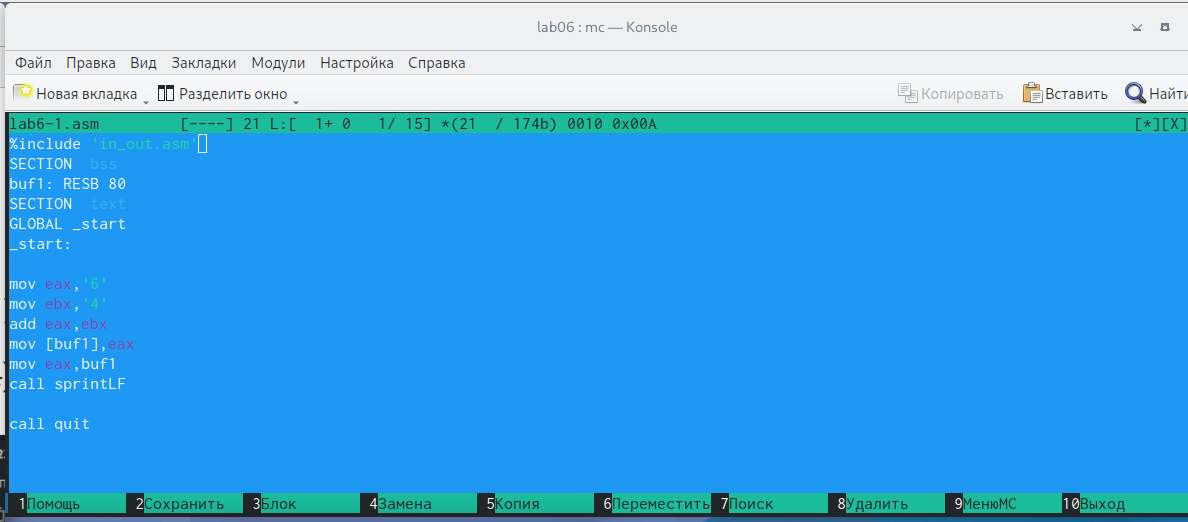
## 4.1 1 Символьные и численные данные в NASM

Перейдём в репозиторий и создадим там файл с пмомщью команды lab6-1.asm.(рис. [??]).



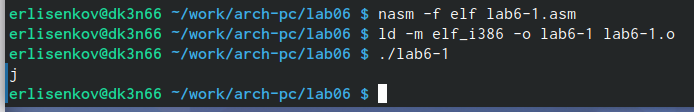
Переход в репозиторий и создание

Выполню перенос файла in\_out.asm в наш репозиторий и начну редактирование файла lab6-1.asm.(рис. [??]).



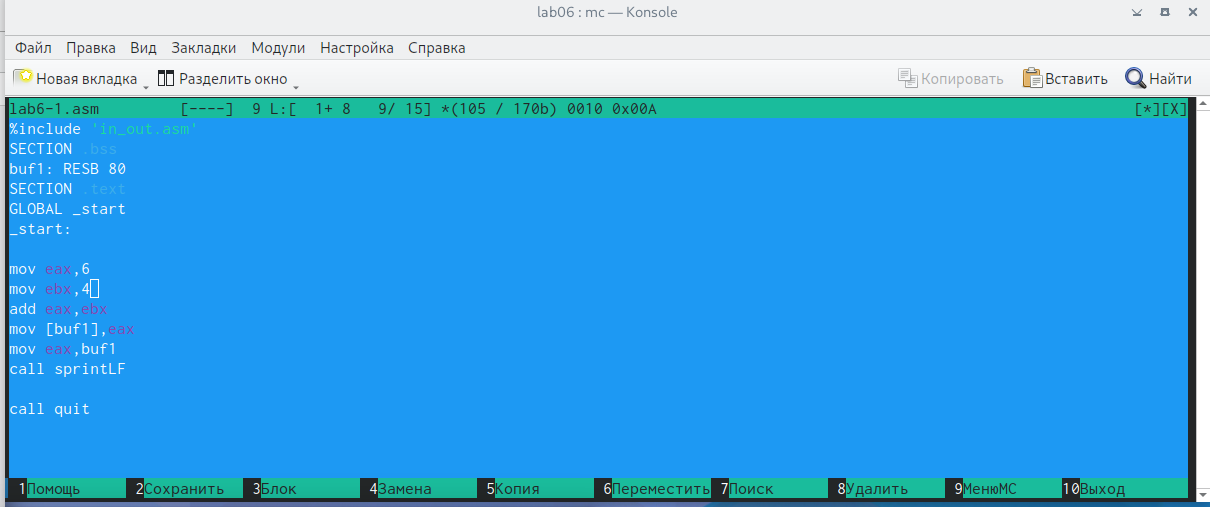
Редактирование файла

Запустим файл. Увидим символ j, потому что программа вывела то, что соответствует по системе ASCII сумме двоичных кодов символов 4 и 6. (рис. [??]).



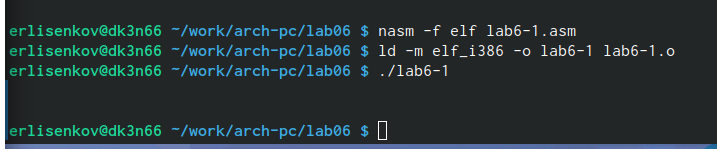
Запуск файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4. (рис. [??]).



Редактирование файла

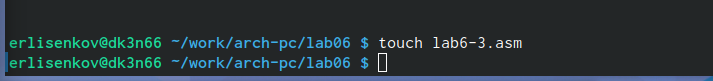
Выполю запуск программы и замечу, что вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран. [??]).



Запуск файла

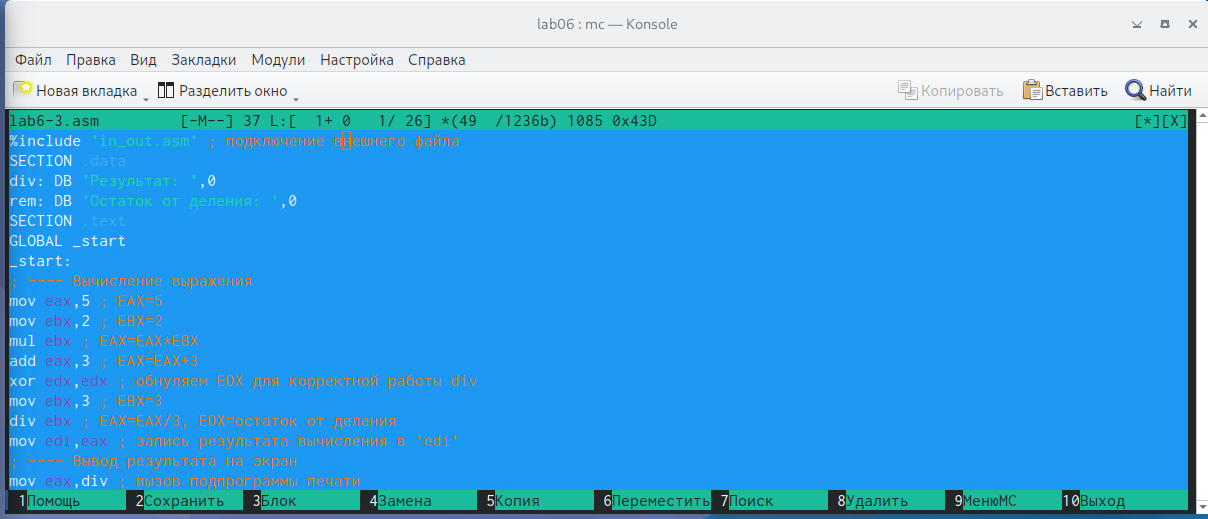
## 4.2 Выполнение арифметических операций в NASMц

Создам файл lab6-3.asm для дальнейших заданий. (рис. [??]).



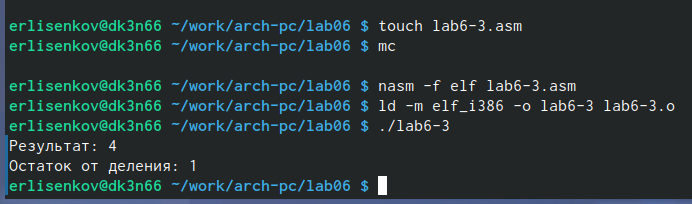
Создам файл

Добавлю новый код для вычисления математической формулы (рис. [??]).



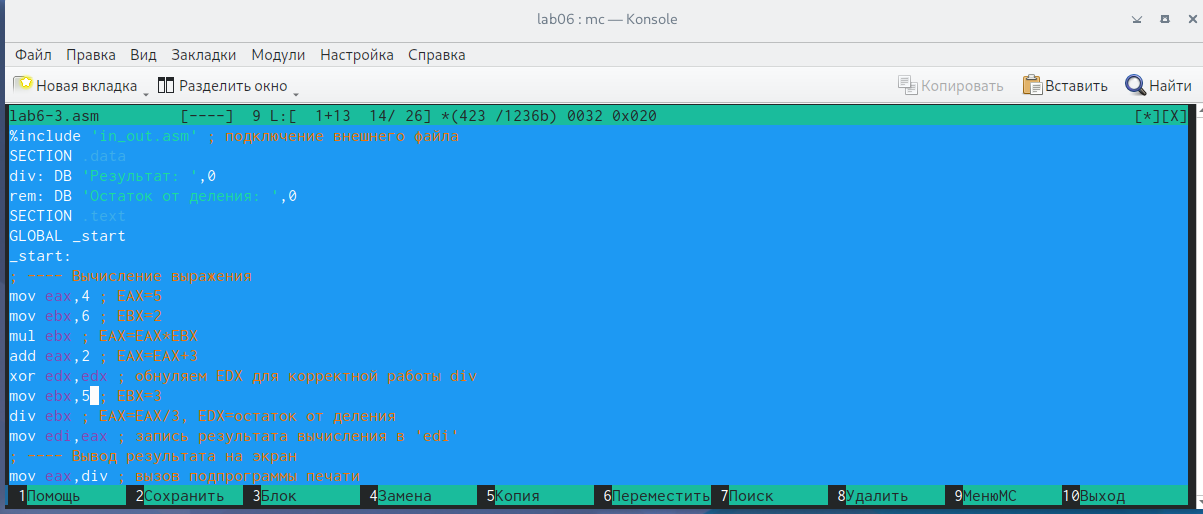
Редактирование файла

Запустим файл и поймём, что всё работает верно (рис. [??]).

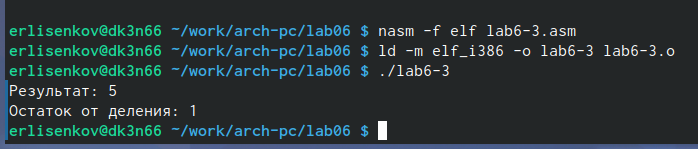


Запуск файла

Отредактирую файл и заново запущу программу, чтобы удостовериться, что всё рабоет верно (рис. [??]) (рис. [??]).



Редактирование файла



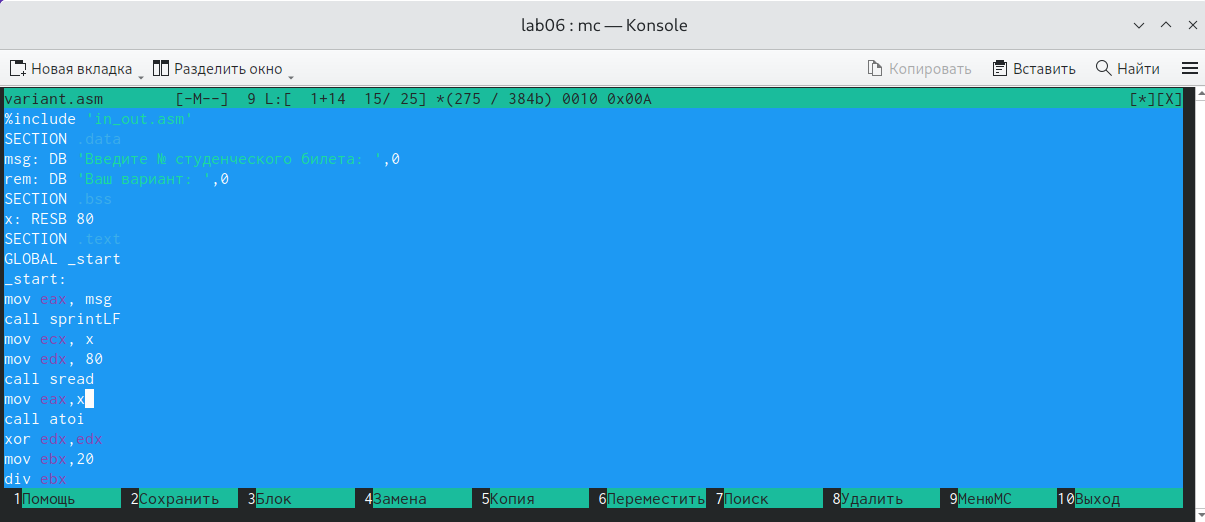
Запуск файла

Создам файл variant (рис. [??]).

Создаю файл

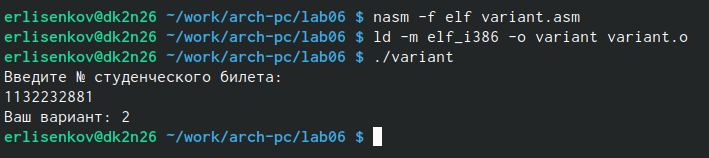
Создаю файл

Выполню редактирование файла (рис. [??]).



Редактирование файла

Запускаю программу (рис. [??]).



Запуск файла

## 4.3 Задание для самостоятельной работы

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. 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 вычисления варианта отвечают строки:

xor edx,edx ; обнуление edx для корректной работы div  
mov ebx,20 ; ebx = 20  
div ebx ; eax = eax/20, edx - остаток от деления  
inc edx ; edx = edx + 1

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 на экран результатов вычислений отвечают строки:

mov eax,edx  
call iprintLF

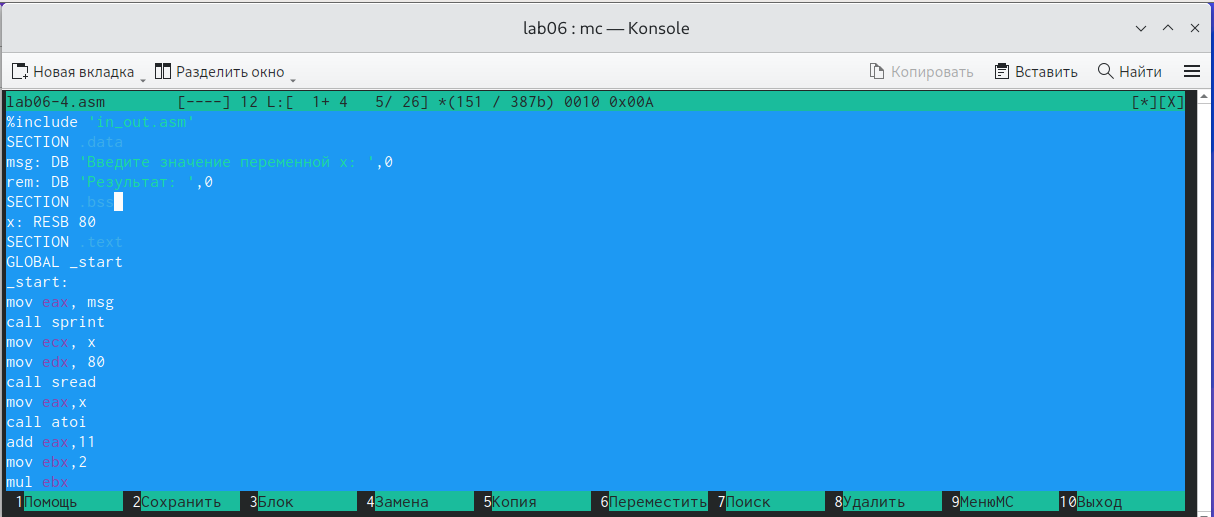
# 5 Выполнение заданий

Создам программу для выполнения задания (рис. [??]).

Создам файл

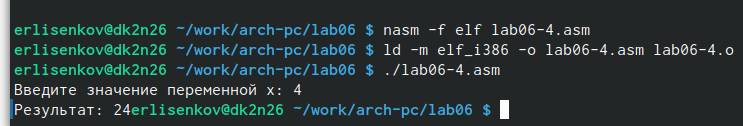
Создам файл

Изменяю файл для корректной работы и вычисления моего задания (рис. [??]).



Редактирование файла

Выполню запуск программы (рис. [??]).



Запуск программы

# 6 Выводы

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

# 7 Список литературы

Лабораторная работа No6. Арифметические операции в NASM.