Отчёт по лабораторной работе №6

Дисциплина: архитектура компьютера

Кузнецова Елизавета Андреевна

Содержание

# 1 Цель работы

Цель лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

# 2 Задание

1. Освоить символьные и численные данные в NASM.
2. Выполнить арифметические операции в NASM.
3. Написать программу вычисления выражения 𝑦 = 𝑓(𝑥). Вариант узнать по программе для студенчекого билета.
4. Загрузить файлы на Github.

# 3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес опе- ранда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. Существует три основных способа адресации: 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.Расширенная таблица ASCII состоит из двух частей. Первая (символы с кодами 0-127) является универсальной, а вторая (коды 128-255) предназначена для специальных символов и букв национальных алфавитов и на компьютерах разных типов может меняться.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 дан- ные 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.

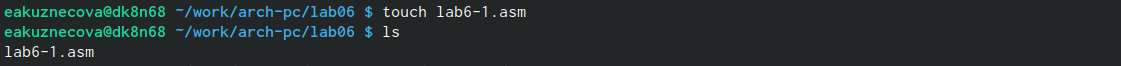
# 4 Выполнение лабораторной работы

С помощью утилиты mkdir создала директорию, в которой буду создавать файлы с программами для лабораторной работы. Перешла в созданный каталог с помощью утилиты cd (рис. [??]).

Создание директории

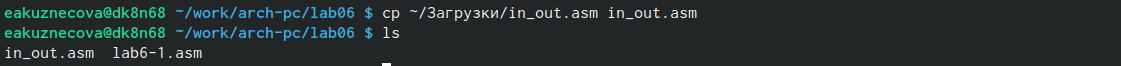
Создание директории

С помощью утилиты touch создала файл lab6-1.asm (рис. [??]).



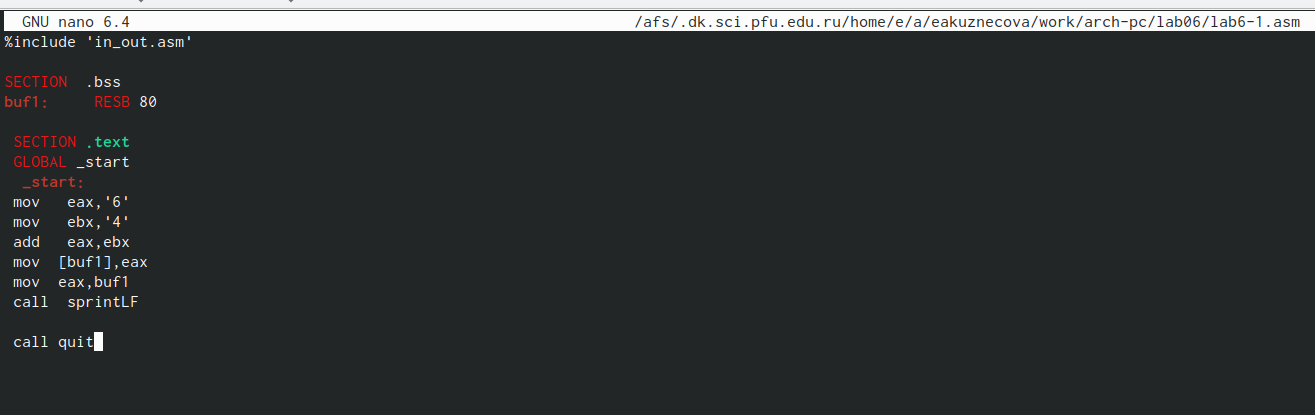
Создание файла

Скопировала в текущий каталог файл in\_out.asm с помощью утилиты cp, так как он будет использоваться в других программах (рис. [??]).



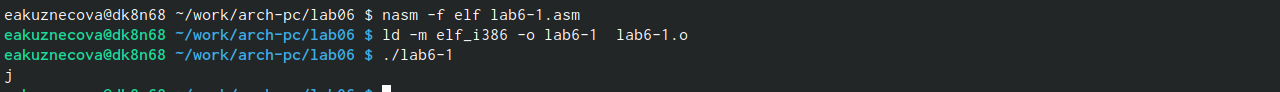
Копирование файла

Открыла созданный файл lab6-1.asm, вставила в него программу вывода значения регистра eax (рис. [??]).



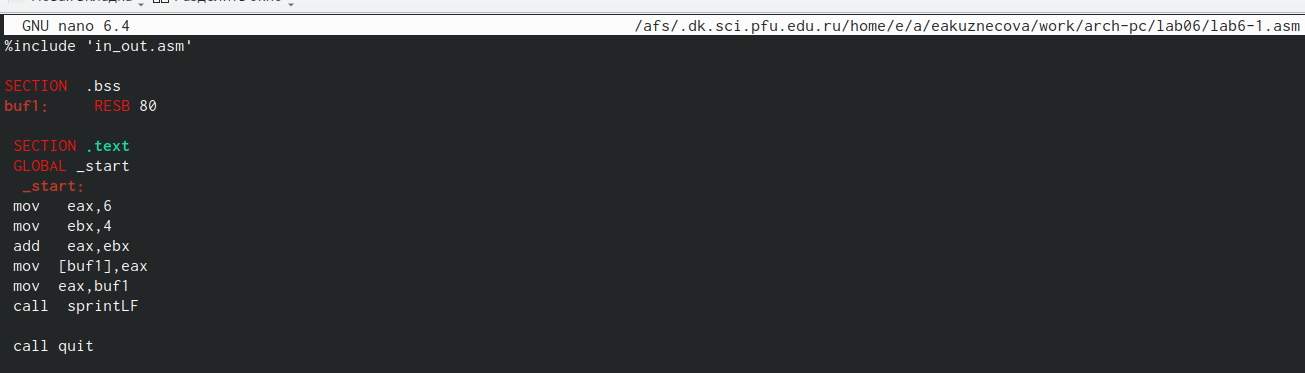
Редактирование файла

Создала исполняемый файл программы и запустила его. Вывелся символ j, потому что программа вывела символ, соответсвующий по система ASCII сумме двоичных кодов и символов 4 и 6 (рис. [??]).



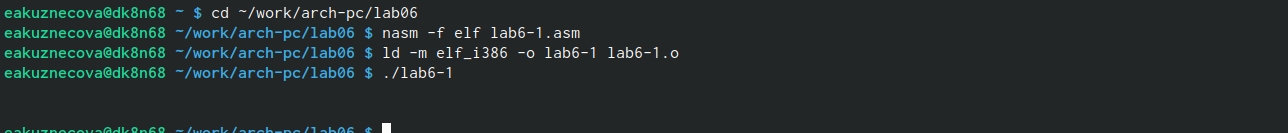
Запуск исполняемого файла

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



Редактирование файла

Создала новый исполняемый файл программы и запустила его. Теперь вывелся символ с кодом 10, это символ перевода строки, он не отображается при выводе на экран (рис. [??]).



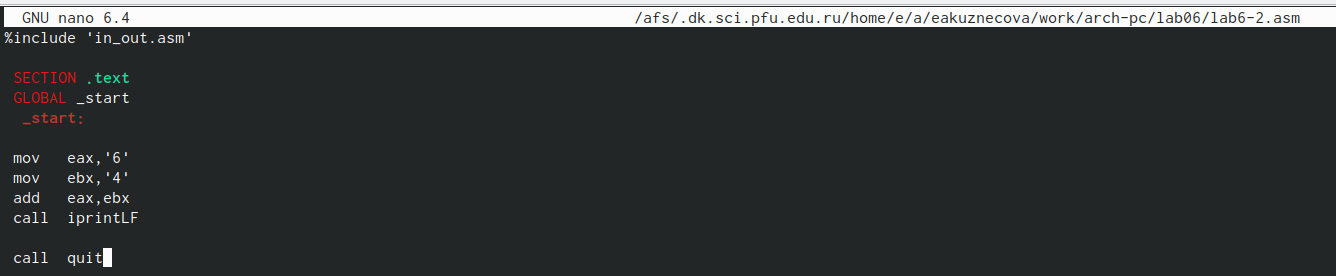
Создание и запуск нового исполняемого файла

Создала новый файл lab6-2.asm с помощью утилиты touch (рис. [??]).

Создание файла

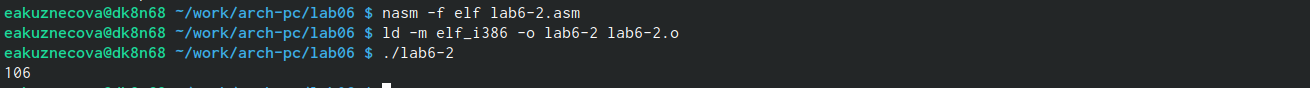
Создание файла

Ввела в файл текст другой программы для вывода значения регистра eax (рис. [??]).



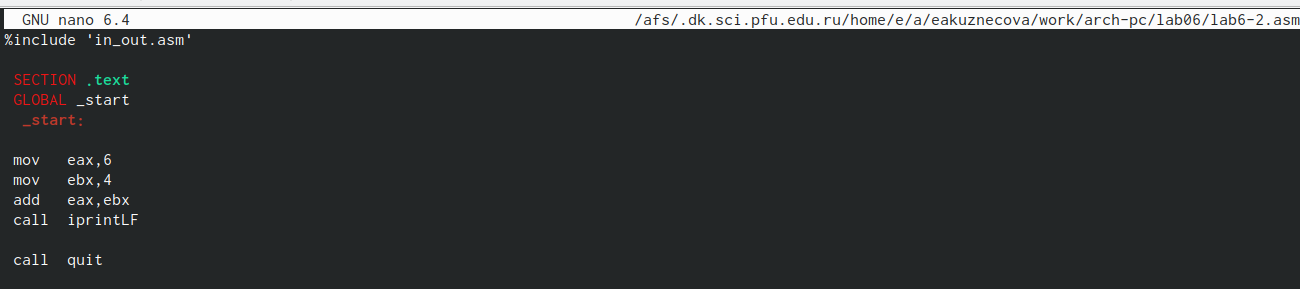
Редакитирование файла

Создала исполняемый файл lab6-2.asm и запустила его. Теперь вывелось число 106, потому что программа позволяет вывести именно число, а не символ, но все равно происходит именно сложение кодов символов “6” и “4”(рис. [??]).



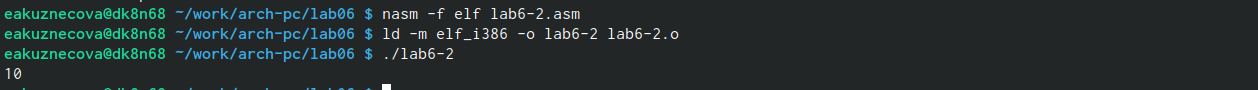
Запуск исполняемого файла

Заменила в тексте программы в файле lab6-2.asm символы “6” и “4” на цифры 6 и 4 (рис. [??]).



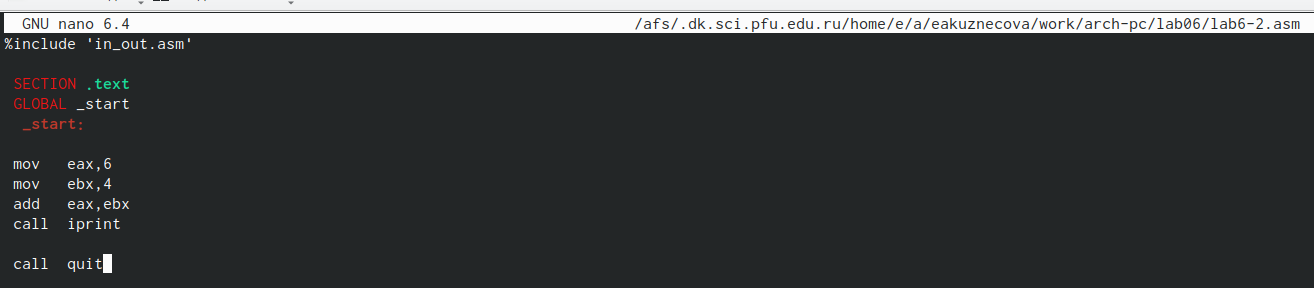
Редактирование файла

Создала новый исполняемый и запустила его. Теперь программа скложила не соответствующие символам коды в системе ASCII, а цифры, поэтому вывод 10 (рис. [??]).



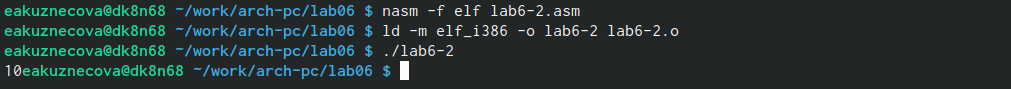
Запуск исполняемого файла

Заменила в тексте программы iprintLF на iprint (рис. [??]).



Редактирование файла

Создала новый исполняемый файл и запусила его. Выводилась также сумма цифр 6 и 4, потому что символ переноса строки не отоброжался, когда программа исполнялась с функцией iprintLF, а iprint не добавляет к выводу символ переноса строки в отличие от iprintLF (рис. [??]).



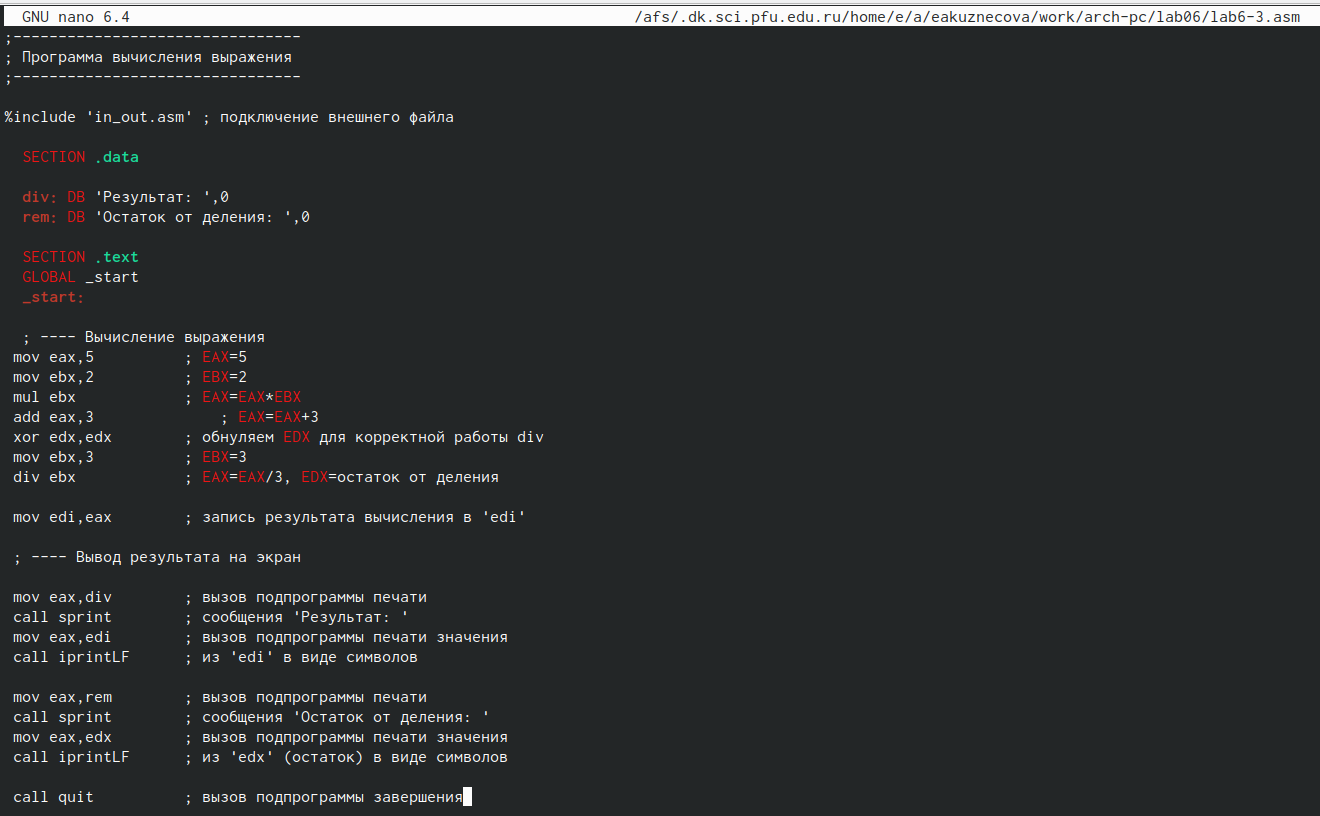
Запуск исполняемого файла

Создала файл lab6-3.asm c помощью утилиты touch (рис. [??]).

Создание файла

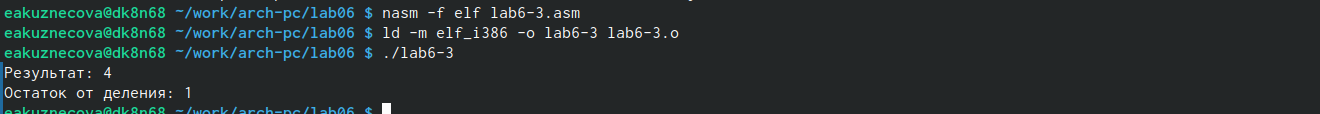
Создание файла

Ввела в созданный файл текст программы для вычисления значения выражения f(x) = (5 \* 2 + 3)/3 (рис. [??]).



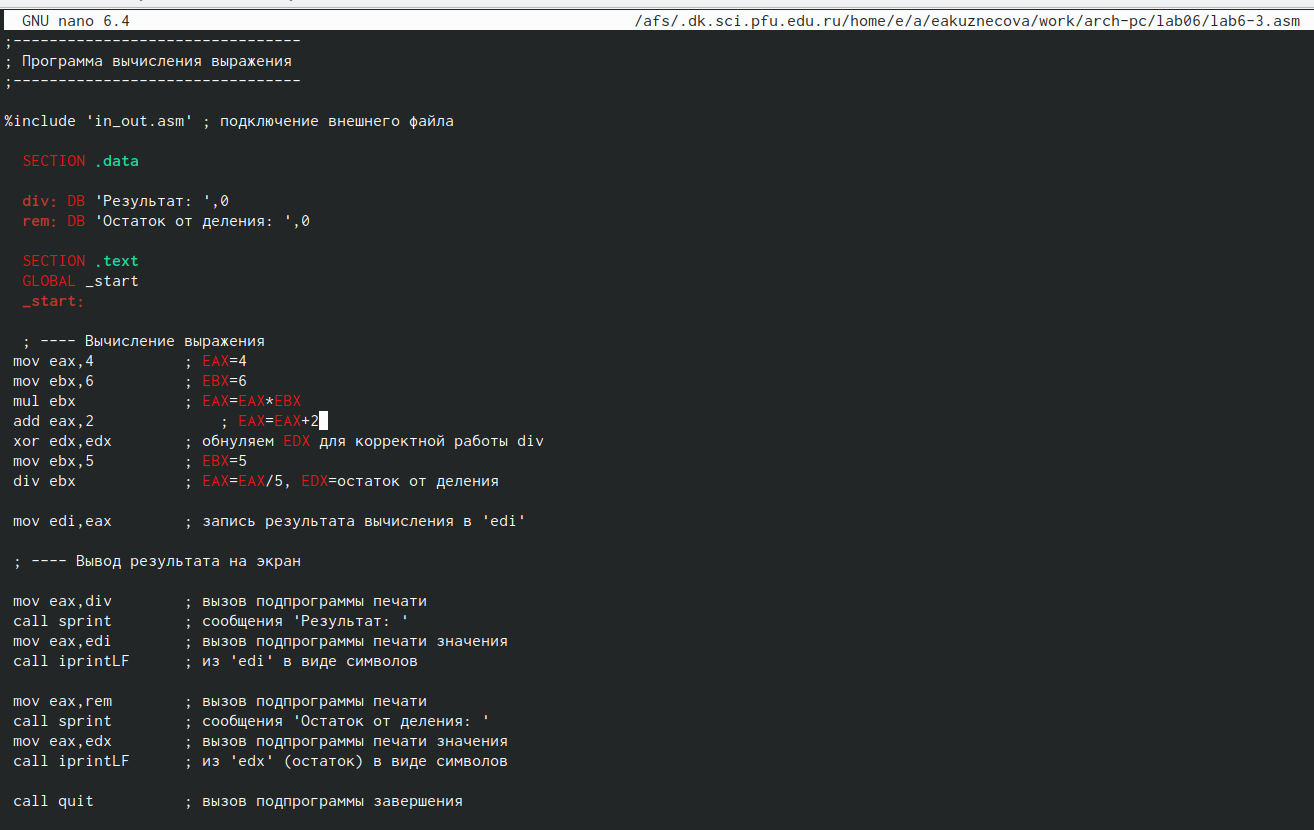
Редактирование файла

Создала исполняемый файл и запустила его (рис. [??]).



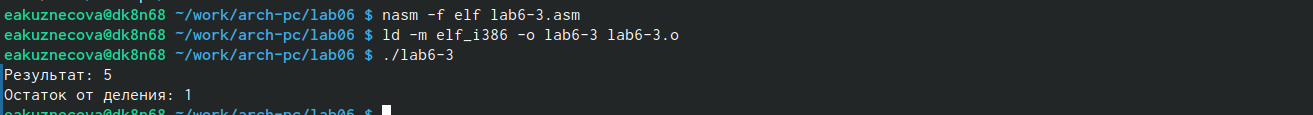
Запуск исполняемого файла

Изменила программу, чтобы она вычисляла значение выражения f(x) = (4 \* 6 + 2)/5 (рис. [??]).



Редактирование программы

Создала новый исполняемый файл и запустила его. Программа правильно была выполнена, я пеерепроверяла самостоятельно (рис. [??]).



Запуск исполняемого файла

Создала файл variant.asm с помощью утилиты touch (рис. [??]).

Создание файла

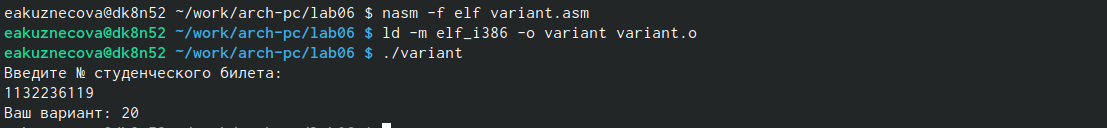
Создание файла

Ввела в созданный файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. [??]).



Редактирование файла

Создала новый исполняемый файл и запустила его. Ввела номер своего студенческого билета с клавиатуры, программа вывела, что мой вариант-20 (рис. [??]).



Запуск исполняемого файла и вывод номера моего варианта

# 5 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. 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   
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 на экран результатов вычислений отвечают строки:

mov eax,edx  
call iprintLF

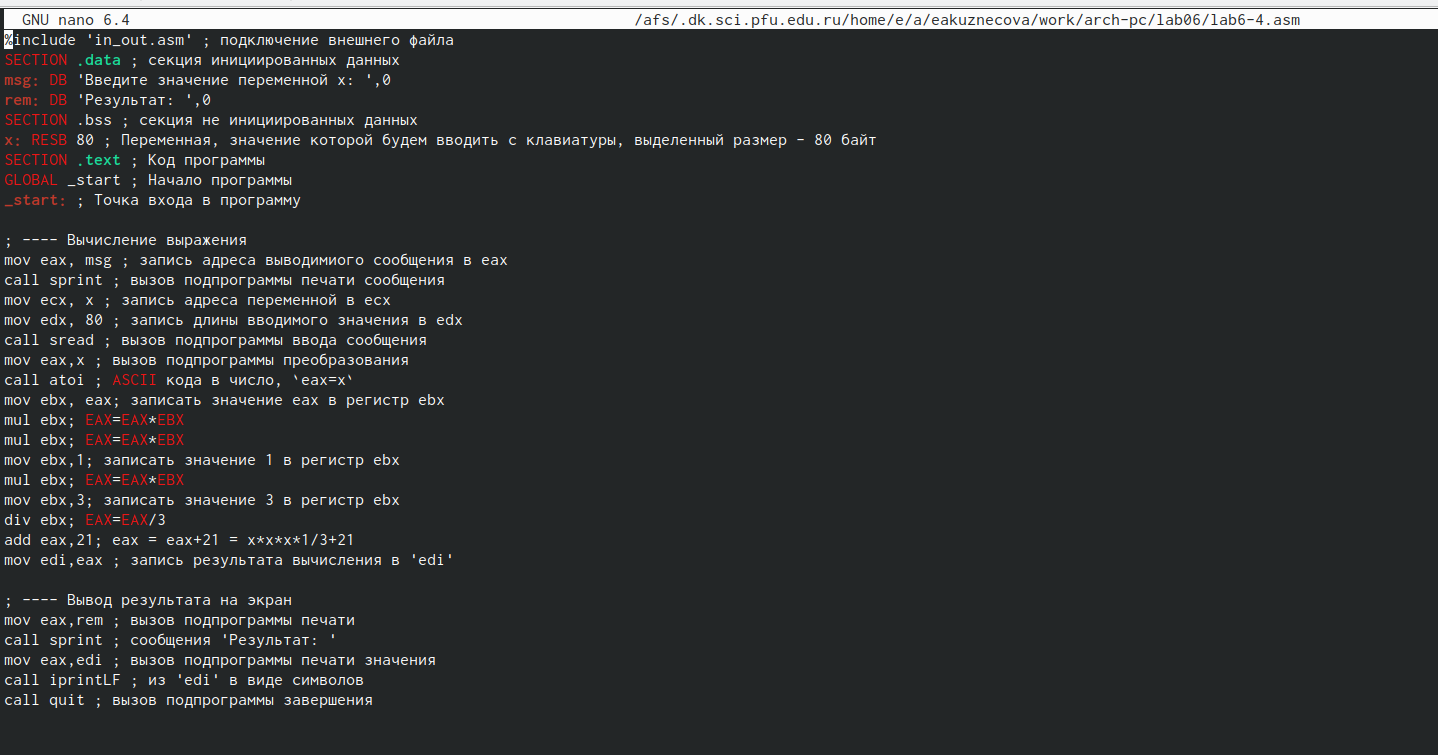
# 6 Выполнение заданий для самостоятельной работы

Создала файл lab6-4.asm с помощью утилиты touch (рис. [??]).

Создание файла

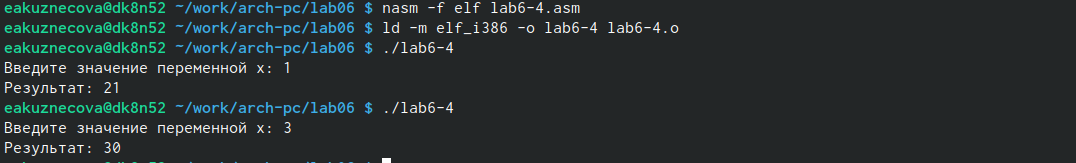
Создание файла

Открыла созданный фвйл и ввела в него текст программы для вычисления значения выражения 𝑥^3\*1/3+21, которое было под 20 вариантом (рис. [??]).



Написание и редактирование программы

Создала новый исполняемый файл и запустила его. При вводе значения 1, выводом является число 21. При вводе значения 3, выводом является число 30. Программа сработала верно, я проверяла самостоятельно (рис. [??]).



Запуск исполняемого файла

# 7 Программа для вычисления значения выражения x^3\*1/3+21

%include 'in\_out.asm' ; подключение внешнего файла  
SECTION .data ; секция инициированных данных  
msg: DB 'Введите значение переменной х: ',0  
rem: DB 'Результат: ',0  
SECTION .bss ; секция не инициированных данных  
x: RESB 80 ; Переменная, значение которой будем вводить с клавиатуры, выделенный размер - 80 байт  
SECTION .text ; Код программы  
GLOBAL \_start ; Начало программы  
\_start: ; Точка входа в программу  
  
; ---- Вычисление выражения  
mov eax, msg ; запись адреса выводимиого сообщения в eax  
call sprint ; вызов подпрограммы печати сообщения  
mov ecx, x ; запись адреса переменной в ecx  
mov edx, 80 ; запись длины вводимого значения в edx  
call sread ; вызов подпрограммы ввода сообщения  
mov eax,x ; вызов подпрограммы преобразования  
call atoi ; ASCII кода в число, `eax=x`  
mov ebx, eax; записать значение eax в регистр ebx  
mul ebx; EAX=EAX\*EBX  
mul ebx; EAX=EAX\*EBX  
mov ebx,1; записать значение 1 в регистр ebx  
mul ebx; EAX=EAX\*EBX  
mov ebx,3; записать значение 3 в регистр ebx  
div ebx; EAX=EAX/3  
add eax,21; eax = eax+21 = x\*x\*x\*1/3+21  
mov edi,eax ; запись результата вычисления в 'edi'  
  
; ---- Вывод результата на экран  
mov eax,rem ; вызов подпрограммы печати  
call sprint ; сообщения 'Результат: '  
mov eax,edi ; вызов подпрограммы печати значения  
call iprintLF ; из 'edi' в виде символов  
call quit ; вызов подпрограммы завершения

# 8 Выводы

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