Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Ермакова Анастасия Алексеевна

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

Цель данной лабораторной работы - освоить арифметические инструкции языка ассемблера NASM.

# 2 Задание

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 опе- ранда 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 ко- манде, Например: mov ax,2. • Адресация памяти – операнд задает адрес в памяти. В команде указывается символи- ческое 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 (см. Приложение.), 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). По- этому, 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

## 4.1 Символьные и численные данные в NASM

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

Рис. 1: Создание каталога для лабораторной работы

Рис. 1: Создание каталога для лабораторной работы

Перехожу в созданный каталог и создаю файл lab6-1.asm. (рис. 2)

Рис. 2: Создание файла lab6-1.asm

Рис. 2: Создание файла lab6-1.asm

Открываю файл с помощью текстового редактора mousepad и ввожу текст программы из листинка 6.1. Создаю исполняемый файл и запускаю его (рис. 3) Программа вывела символ j, т.к. он соответствует сумме двоичных кодов символов 4 и 6 по системе ASCII.

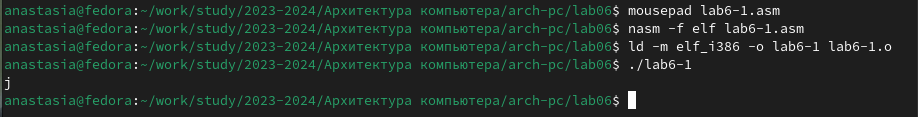


Рис. 3: Создание исполняемого файла. Результат его работы

Вношу изменения в текст программы, вместо символов записываю регистры числа, заменяю ‘6’ и ‘4’ на 6 и 4. (рис. 4)

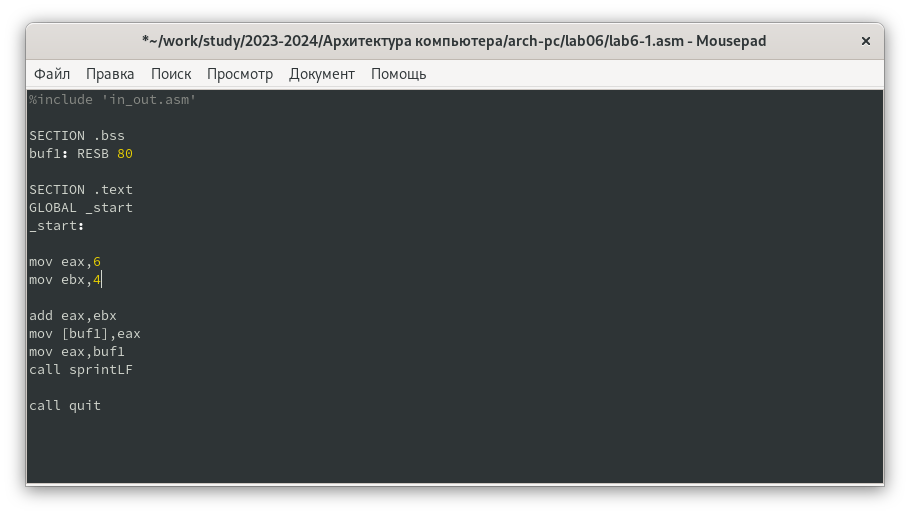


Рис. 4: Измененный текст программы

Создаю исполняемый файл и запускаю его (рис. 5) Результатом работы является символ с кодом 10, то есть символ перевода строки. На экран этот символ не отображается.

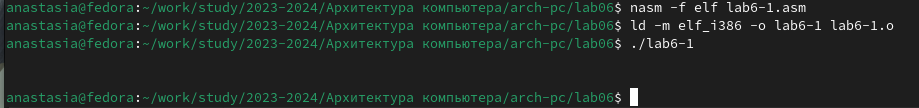


Рис. 5: Результат работы исполняемого файла

Создаю файл lab6-2.asm и открывю его с помощью текстового редактора. (рис. 6)

Рис. 6: Создание файла lab6-2.asm

Рис. 6: Создание файла lab6-2.asm

Ввожу в него текст программы из листинга 6.2. (рис. 7)

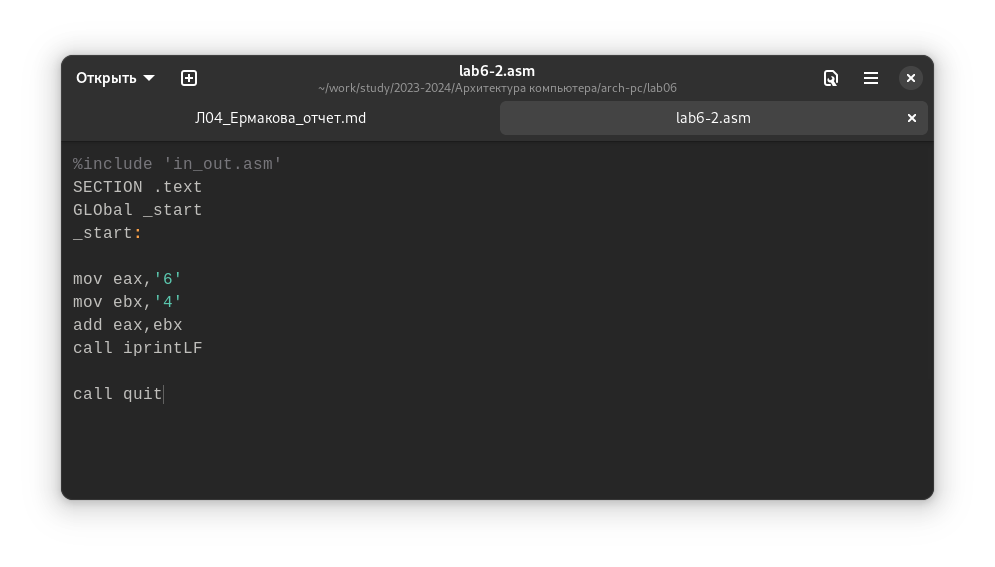


Рис. 7: Текст программы

Создаю исполняемый файл и запускаю его (рис. 8) Программа вывела число 106, потому что она позволяет вывести именно число, а не код, после сложения символов ‘6’ и ‘4’.

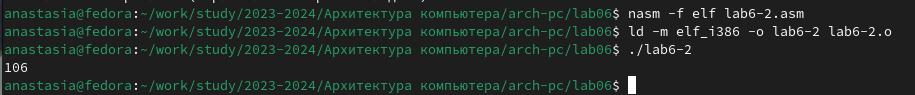


Рис. 8: Результат работы исполняемого файла

Изменяю текст программы, заменяю ‘6’ и ‘4’ на числа 6 и 4. (рис. 9)

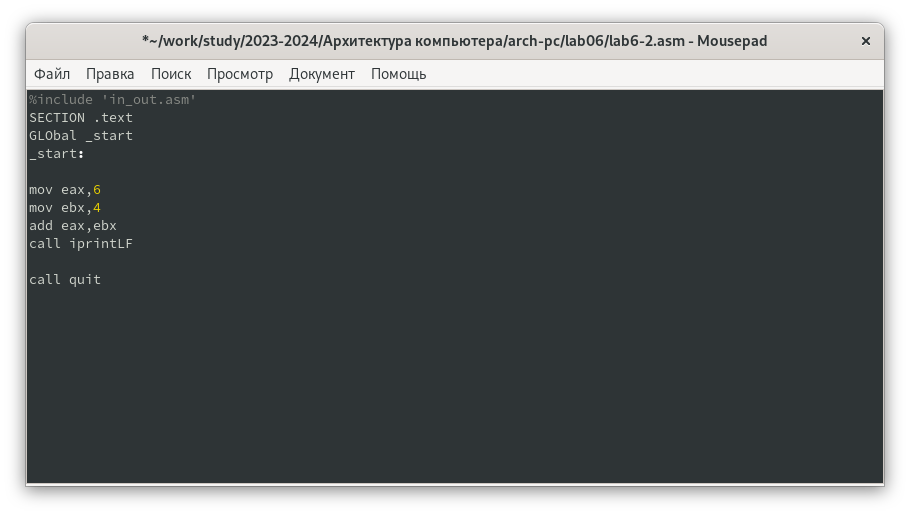


Рис. 9: Измененный текст программы

Создаю исполняемый файл и запускаю его (рис. 10) Программа сложила 6 и 4, поэтому выводит число 10.

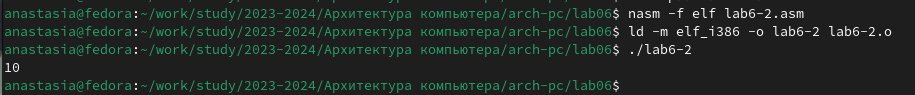


Рис. 10: Результат работы программы

Заменяю в тексте программы функцию iprintLF на iprint. (рис. 11) Функция iprint не добавляет к выводу символ переноса строки, в отличие от функции iprintLF, поэтому после числа 10 нет переноса.

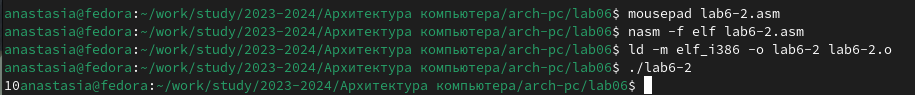


Рис. 11: Результат работы измененного файла

## 4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch и ввожу в него текст программы из листинга 6.3 (рис. 12)

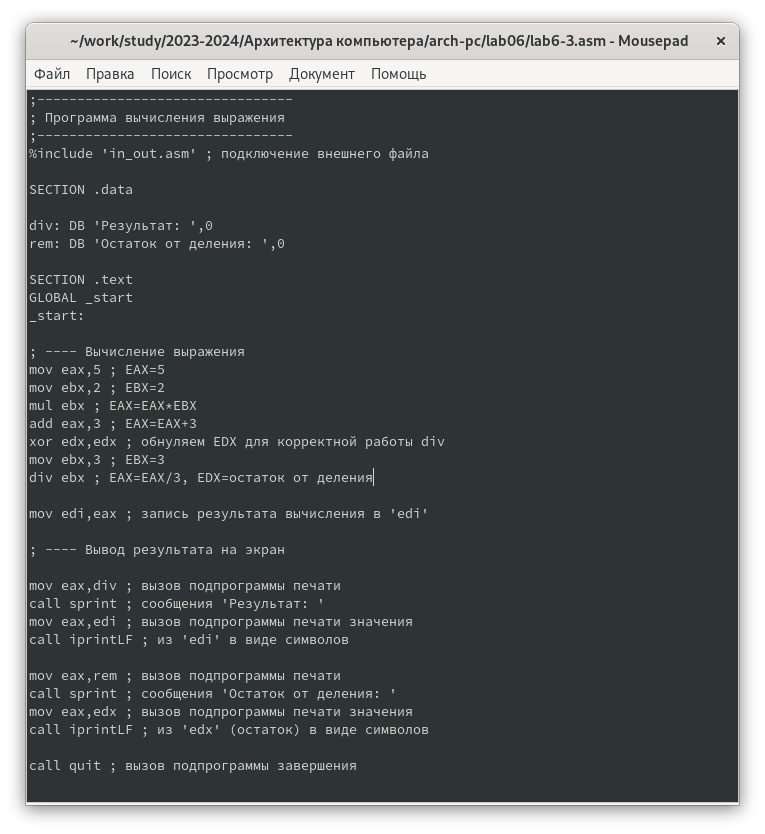


Рис. 12: Текст программы файла lab6-3.asm

Создаю исполняемый файл и запускаю его (рис. 13) Результат работы программы совпадает с тем, что представлен в файле на ТУИС.

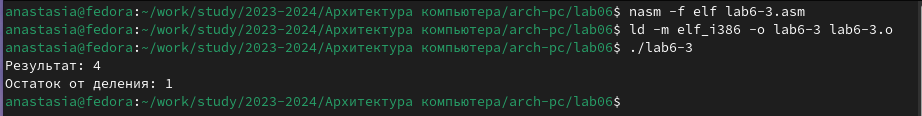


Рис. 13: Результат работы программы

Вношу изменения в текст программы, чтобы она посчитала значение выражения (4\*6 + 2)/5 (рис. 14)

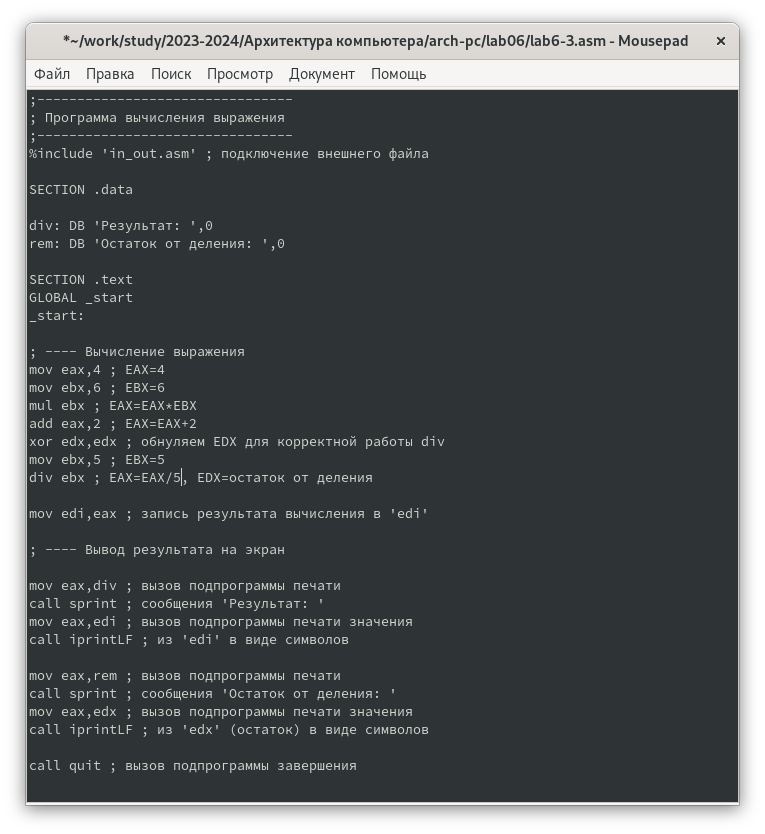


Рис. 14: Измененный текст программы

Создаю исполняемый файл и запускаю его (рис. 15) Результатом является 5, а остатком от деления 1. Программа работает верно.

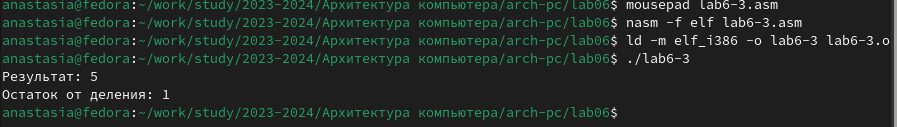


Рис. 15: Результат работы измененной программы

Далее создаю файл variant.asm и ввожу в него текст программы из листинга 6.4 (рис. 16)

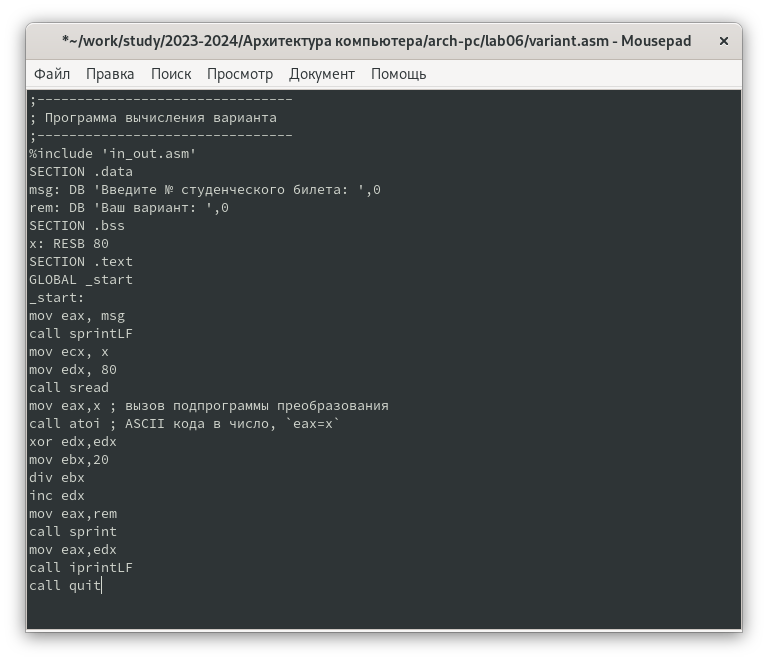


Рис. 16: Текст программы файла variant.asm

Создаю исполняемый файл и запускаю его (рис. 17) Программа вычислила вариант задания по номеру студенческого билета, который я ввела с клавиатуры. На экран вывелся мой вариант (19).

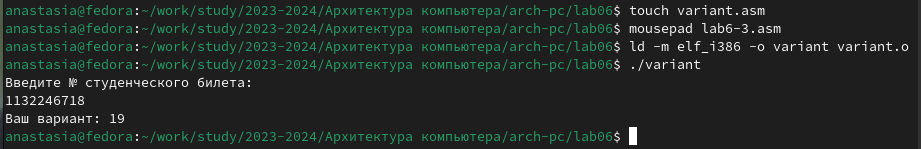


Рис. 17: Результат работы программы

### 4.2.1 Ответы на вопросы

1. За вывод сообщения “Ваш вариант” отвечают следующие строки кода:

mov eax,rem  
call sprint

1. Инструкция mov ecx, x используется для того, чтобы положить адрес вводимой строки х в регистр ecx. Инструкция mov edx,80 используется для того, чтобы записать в регистр edx длины вводимой строки. Инструкция call sread - для вызова подпрограммы из внешнего файла для ввода сообщения с клавиатуры.
2. Инструкция call atoi используется для вызова подпрограммы из внешнего файла для преобразования ASCII-кода символа в целое число и записывает результат в регистр eax.
3. За вычисление варианта отвечают следующие 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 следующие строки:

mov eax,edx  
call sprintLF

## 4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm с помощью утилиты touch и открываю файл для редактирования. (рис. 18)

Рис. 18: Создание файла lab6-4.asm

Рис. 18: Создание файла lab6-4.asm

Ввожу в созданный файл текст программы для вычисления выражения под номером 19: (1/3*x + 5)*7 (рис. 19)

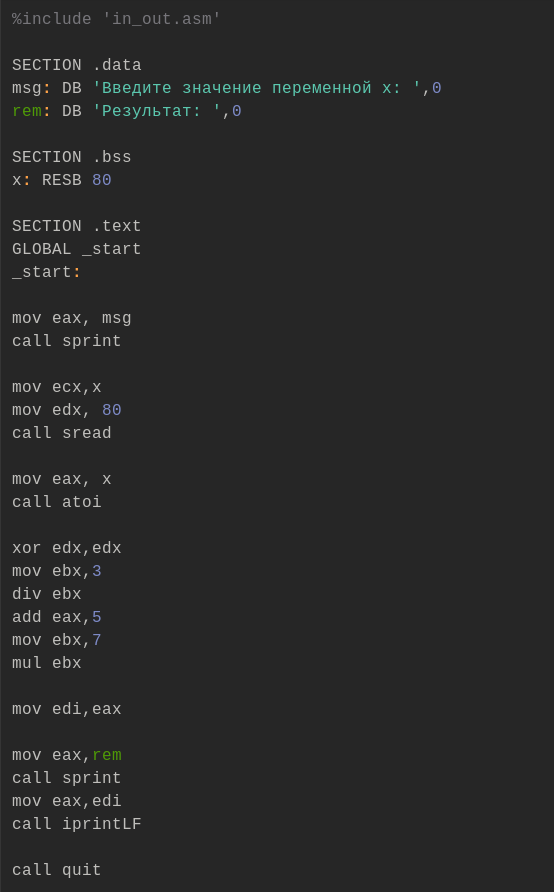


Рис. 19: Текст программы файла lab6-4.asm

Создаю исполняемый файл и запускаю его (рис. 20) Ввожу с клавиатуры значение переменной 3, результат выводится на экран - ответ 42. Повторно запускаю программу и ввожу 9, результатом является 56. Программа работает исправно, ответы верные.

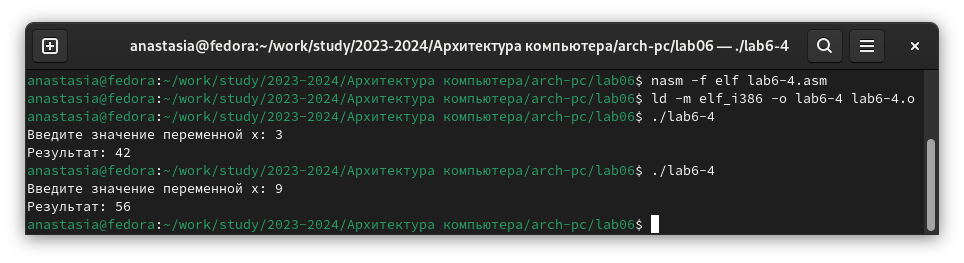


Рис. 20: Результат работы программы

Код программы для вычисления значения выражения (1/3*x + 5)*7:

%include 'in\_out.asm' ; подключение внешнего файла  
  
SECTION .data ; секция инициированных данных  
msg: DB 'Введите значение переменной x: ',0  
rem: DB 'Результат: ',0  
  
SECTION .bss ; секция не инициированных данных  
x: RESB 80 ; переменная, значение которой будем вводить с клавиатуры  
  
SECTION .text ; код программы  
GLOBAL \_start ; начало программы  
\_start: ; точка входа в программу  
  
mov eax, msg ; запись адреса вводимого сообщения в eax  
call sprint ; вызов подпрограммы печати сообщения  
  
mov ecx,x ; запись адреса переменной х в есх  
mov edx, 80 ; запись длинны вводимого сообщения в edx  
call sread ; вызов подпрограммы ввода сообщения  
  
mov eax, x ; вызов подпрограммы преобразования  
call atoi ; ASCII кода в число 'eax=x'  
  
xor edx,edx ; обнуление edx для корректной работы div  
mov ebx,3 ; ebx = 3  
div ebx ; eax = eax/3, edx - остаток от деления  
add eax,5 ; eax = eax+5  
mov ebx,7 ; ebx = 7  
mul ebx ; eax = eax\*ebx   
  
mov edi,eax ; запись результата вычисления в 'edi'  
  
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

# 5 Выводы

В ходе выполнения лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

# 6 Список литературы

1. [Архитектура ЭВМ](https://esystem.rudn.ru/mod/resource/view.php?id=1030554)