Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Ганина Таисия, НКАбд-01-22

Содержание

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

# 2 Задание

1. Создать файл lab7-1.asm и ввести в него программу из листинга 1, создать исполняемый файл и запустить его.
2. Исправить листинг 1, заменив строки

mov eax,'6'

mov ebx,'4'

на строки

mov eax,6

mov ebx,4

Создать исполняемый файл и запустить его, пользуясь таблицей ASCII определить какому символу соответствует код 10.

1. Создать файл lab7-2.аsm, ввести в него программу из листинга 2, создать исполняемый файл и запустить его.
2. Исправить листинг 2, заменив строки

mov eax,'6'

mov ebx,'4'

на строки

mov eax,6

mov ebx,4

Создать исполняемый файл и запустить его.

1. Заменить функцию iprintLF на iprint. Создать исполняемый файл и запустить его. Выяснить чем отличается вывод функций iprintLF и iprint.
2. Создать файл lab7-3.asm, заполнить его соответственно с листингом 3, создать исполняемый файл и запустить его.
3. Изменить файл так, чтобы программа вычисляла выражение
4. Создать файл “вариант”, заполнить его соответственно с листингом 4, создать исполняемый файл и запустить его.
5. Ответить на вопросы по разделу.
6. Написать программу для вычисления выражения и проверить его при х=2 и при х=3.

# 3 Теоретическое введение

Схема команды целочисленного сложения add *(от англ. addition - добавление)* выполняет сложение двух операндов и записывает результат по адресу первого операнда.

1. Команда add работает как с числами со знаком, так и без знака и выглядит следующим образом:

add <операнд\_1>, <операнд\_2>

Допустимые сочетания операндов для команды add аналогичны сочетаниям операндов для команды mov. Так, например, команда add eax,ebx прибавит значение из регистра eax к значению из регистра ebx и запишет результат в регистр eax.

1. Команда целочисленного вычитания sub *(от англ. subtraction – вычитание)* работает аналогично команде add и выглядит следующим образом:

sub <операнд\_1>, <операнд\_2>

Так, например, команда sub ebx,5 уменьшает значение регистра ebx на 5 и записывает результат в регистр ebx.

1. Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называется инкрементом, а вычитание — декрементом. Для этих операций существуют специальные команды: inc *(от англ. increment)* и dec *(от англ. decrement)*, которые увеличивают и уменьшают на 1 свой операнд. Эти команды содержат один операнд и имеет следующий вид:

inc <операнд>

dec <операнд>

Операндом может быть регистр или ячейка памяти любого размера. Команды инкремента и декремента выгодны тем, что они занимают меньше места, чем соответствующие команды сложения и вычитания. Так, например, команда inc ebx увеличивает значение регистра ebx на 1, а команда inc ax уменьшает значение регистра ax на 1.

1. Еще одна команда, которую можно отнести к арифметическим командам это команда изменения знака neg:

neg <операнд>

Команда neg рассматривает свой операнд как число со знаком и меняет знак операнда на противоположный. Операндом может быть регистр или ячейка памяти любого размера.

1. Умножение и деление, в отличии от сложения и вычитания, для знаковых и беззнаковых чисел производиться по-разному, поэтому существуют различные команды.

Для беззнакового умножения используется команда mul *(от англ. multiply – умножение)*:

mul <операнд>

Для знакового умножения используется команда imul:

imul <операнд>

Для команд умножения один из сомножителей указывается в команде и должен находиться в регистре или в памяти, но не может быть непосредственным операндом. Второй сомножитель в команде явно не указывается и должен находиться в регистре EAX,AX или AL, а результат помещается в регистры EDX:EAX, DX:AX или AX, в зависимости от размера операнда.

1. Для деления, как и для умножения, существует 2 команды div (от англ. divide - деление) и idiv:

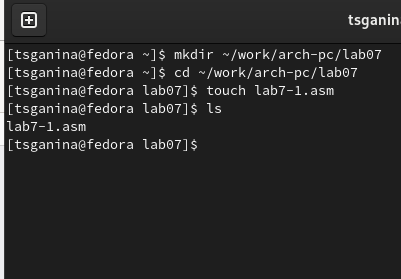
div <делитель>

idiv <делитель>

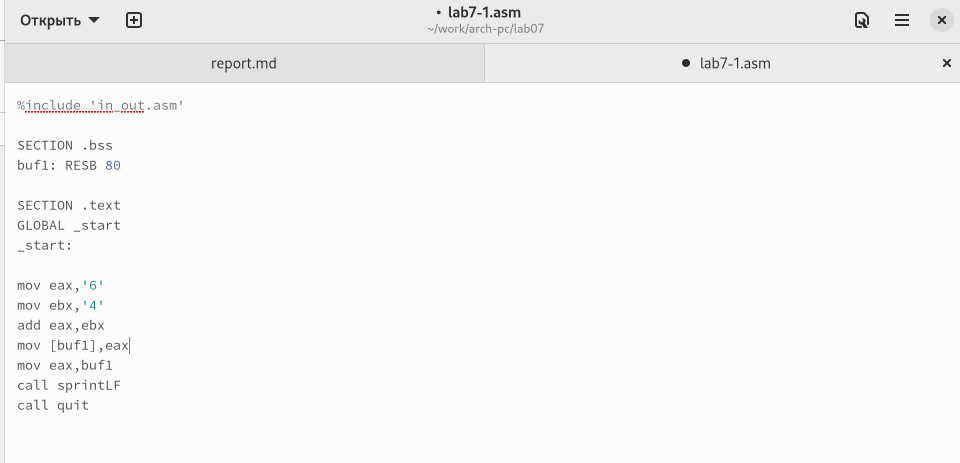
В командах указывается только один операнд – делитель, который может быть регистром или ячейкой памяти, но не может быть непосредственным операндом. Местоположение делимого и результата для команд деления зависит от размера делителя. Кроме того, так как в результате деления получается два числа – частное и остаток, то эти числа помещаются в определённые регистры

# 4 Выполнение лабораторной работы

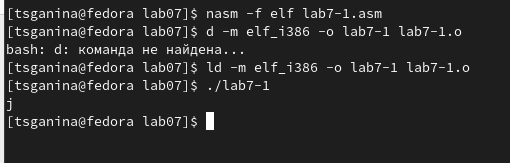
1. Я создала файл lab7-1.asm и ввела в него программу из листинга 1, создала исполняемый файл и запустила его (рис. 1, 2, 3).



1: Рис. 1



2: Рис. 2



3: Рис. 3

1. Исправила листинг 1, заменив строки

mov eax,'6'

mov ebx,'4'

на строки

mov eax,6

mov ebx,4

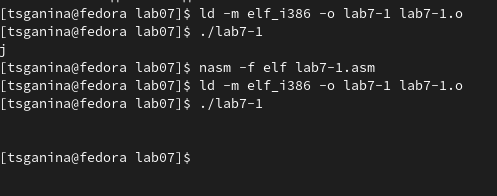
Создала исполняемый файл и запустила его, пользуясь таблицей ASCII определила какому символу соответствует код 10, (рис. 4, 5, 6).



4: Рис. 4



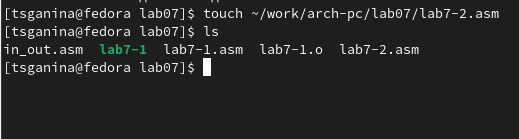
5: Рис. 5



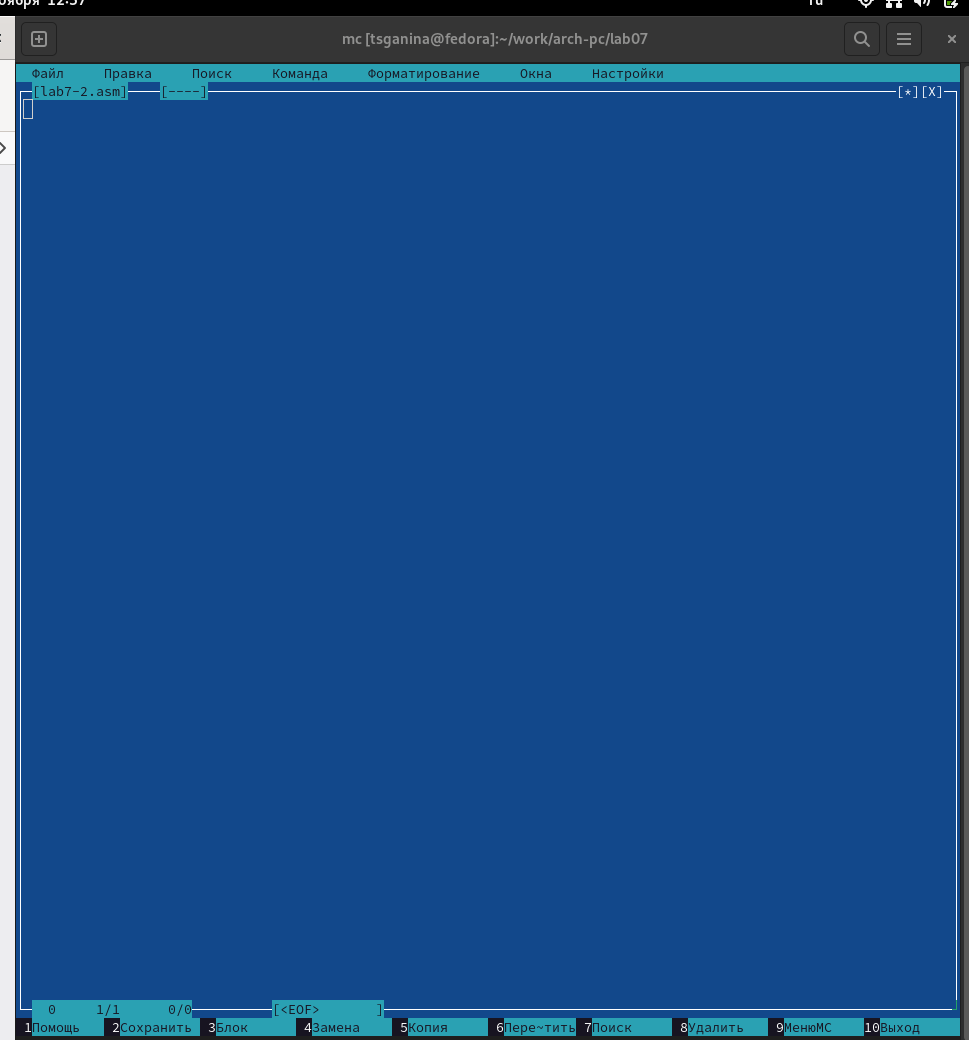
6: Рис. 6

Вывод: код 10 соответствует символу переноса строки, но на экране этот символ не отображается.

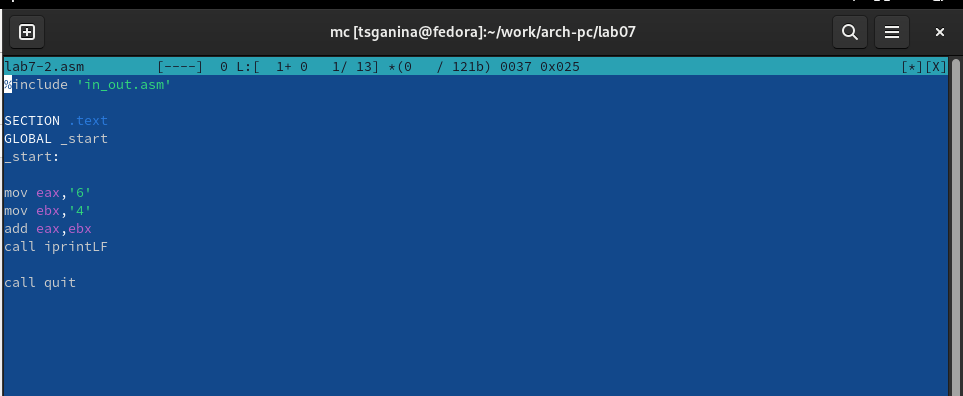
1. Создала файл lab7-2.аsm, ввела в него программу из листинга 2, создала исполняемый файл и запустила его (рис. 7, 8, 9, 10).



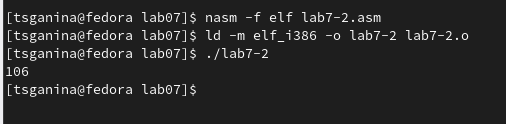
7: Рис. 7



8: Рис. 8



9: Рис. 9



10: Рис. 10

1. Исправила листинг 2, заменив строки

mov eax,'6'

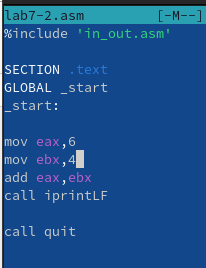
mov ebx,'4'

на строки

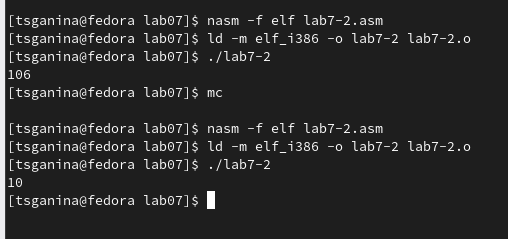
mov eax,6

mov ebx,4

Создала исполняемый файл и запустила его (рис. 11, 12).

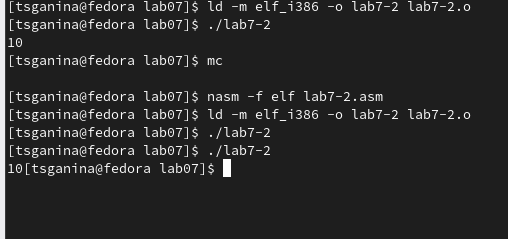


11: Рис. 11



12: Рис. 12

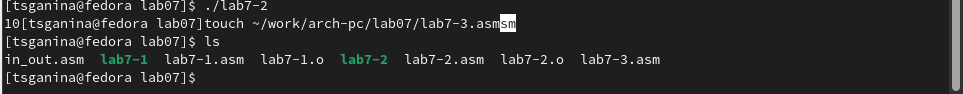
1. Заменила функцию iprintLF на iprint. Создала исполняемый файл и запустила его. Выяснила чем отличается вывод функций iprintLF и iprint (рис. 13).



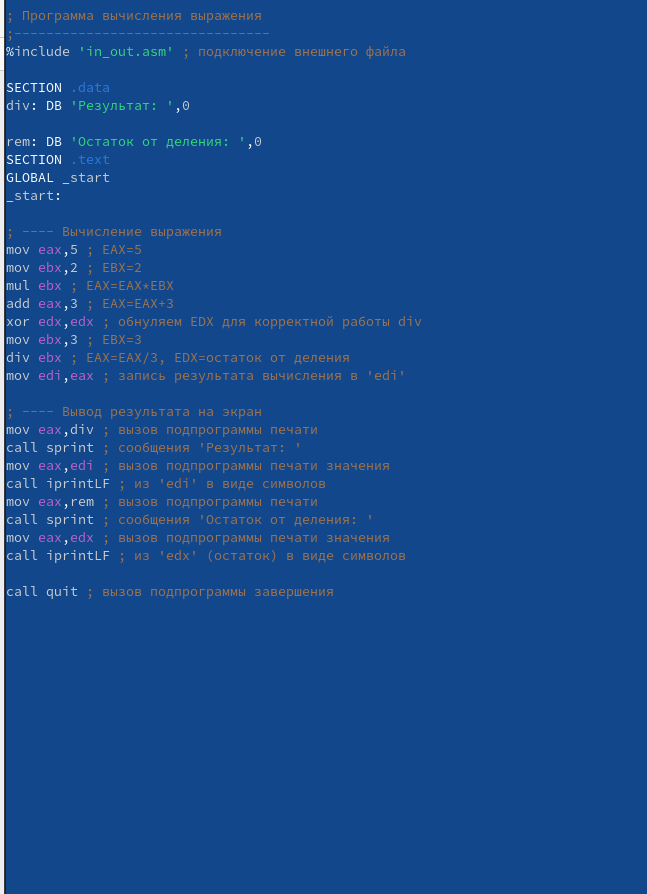
13: Рис. 13

Вывод: отличие состоит в том, что iprint не совершает перенос строки.

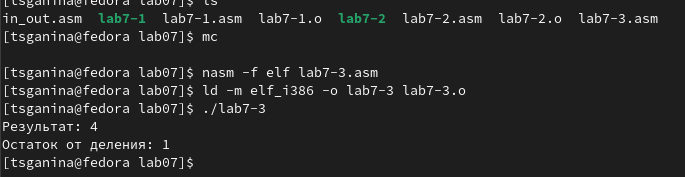
1. Создала файл lab7-3.asm, заполнила его соответственно с листингом 3, создала исполняемый файл и запустила его (рис. 14, 15, 16).



14: Рис. 14

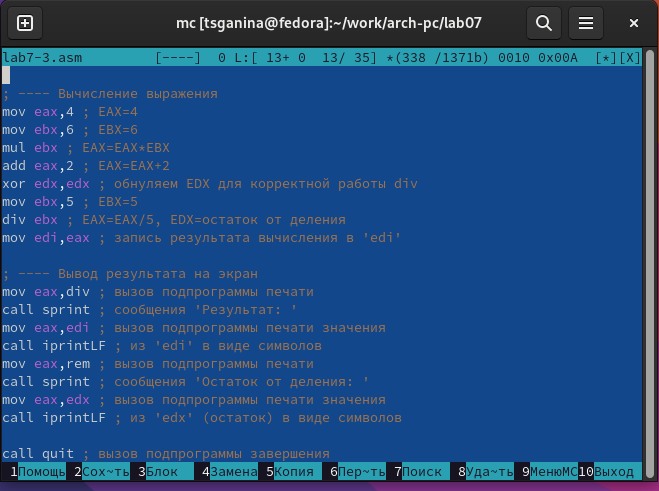


15: Рис. 15

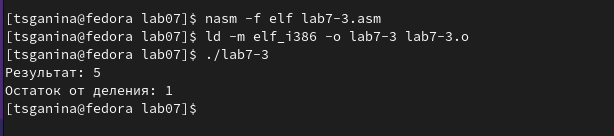


16: Рис. 16

1. Изменила файл так, чтобы программа вычисляла выражение (рис. 17, 18).

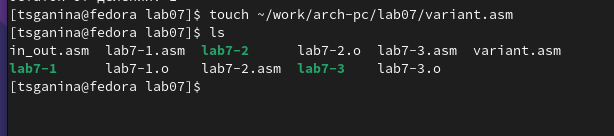


17: Рис. 17

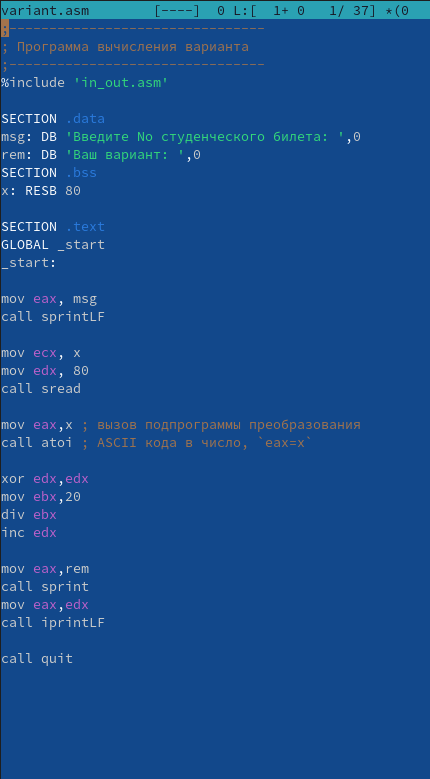


18: Рис. 18

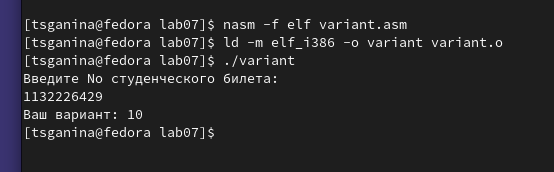
1. Создала файл “вариант”, заполнила его соответственно с листингом 4, создала исполняемый файл и запустила его (рис. 19, 20, 21).



19: Рис. 19



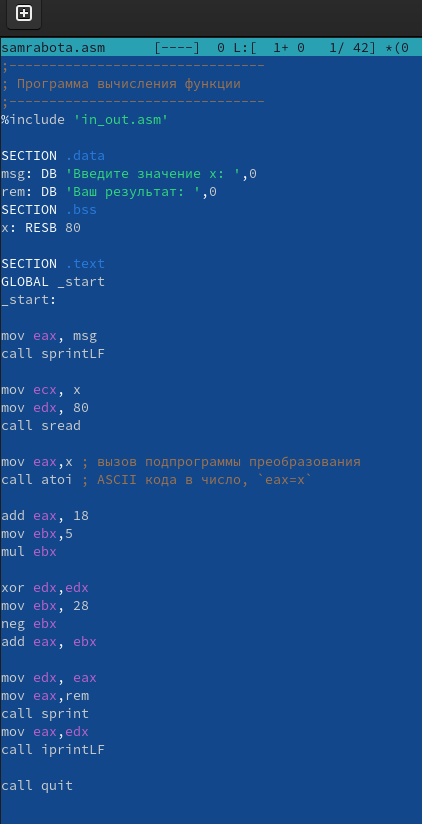
20: Рис. 20



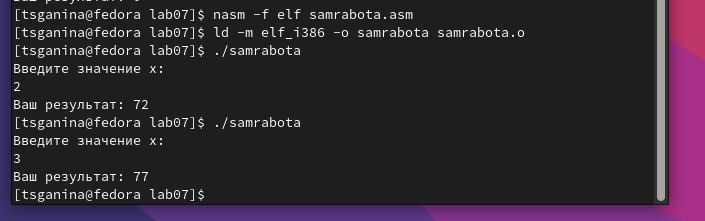
21: Рис. 21

Выполнив те же вычисления вручную, выяснила, что ответ, данный программой, верен.

1. Отвечаю на вопросы по разделу:
   1. Какие строки листинга 7.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?
   * `mov eax,rem`  
       
      `call sprint`
   1. Для чего используется следующие инструкции?
   * `mov ecx, x` - адрес вводимой строки `x` записывется в регистр `ecx`.  
       
      `mov edx, 80` - 80 - длина вводимой строки, записана в `edx`.  
       
      `call sread` - считывание ввода с клавиатуры.
   1. Для чего используется инструкция “call atoi”?
   * Эта инструкция вызывает программу из файла "in\_out.asm" и преобразует   
      ascii-код символа в целое число и записает результат в регистр eax.
   1. Какие строки листинга 7.4 отвечают за вычисления варианта?
   * xor edx,edx
   * mov ebx,20
   * div ebx
   * inc edx
   1. В какой регистр записывается остаток от деления при выполнении ин- струкции “div ebx”?
   * В регистр `edx`
   1. Для чего используется инструкция “inc edx”?
   * Для того, чтобы прибавить к значению `edx` единицу
   1. Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений?
   * mov eax,edx
   * call iprintLF
2. Написать программу для вычисления выражения и проверить его при х=2 и при х=3 (рис. 22, 23).



22: Рис. 22



23: Рис. 23

Проверила себя, выполнив вычисления вручную - ответ получен верный.

# 5 Листинги программ:

1. lab7-1.asm

* %include 'in\_out.asm'  
    
   SECTION .bss  
   buf1: RESB 80  
    
   SECTION .text  
   GLOBAL \_start  
   \_start:  
    
   mov eax,6  
   mov ebx,4  
   add eax,ebx  
   mov [buf1],eax  
   mov eax,buf1  
   call sprintLF  
    
   call quit

1. lab7-2.asm

* %include 'in\_out.asm'  
    
   SECTION .text  
   GLOBAL \_start  
   \_start:  
    
   mov eax,6  
   mov ebx,4  
   add eax,ebx  
   call iprint  
    
   call quit

1. lab7-3.asm

* ;--------------------------------  
   ; Программа вычисления выражения  
   ;--------------------------------  
   %include 'in\_out.asm' ; подключение внешнего файла  
    
   SECTION .data  
   div: DB 'Результат: ',0  
    
   rem: DB 'Остаток от деления: ',0  
   SECTION .text  
   GLOBAL \_start  
   \_start:  
    
   ; ---- Вычисление выражения  
   mov eax,4 ; EAX=4  
   mov ebx,6 ; EBX=6  
   mul ebx ; EAX=EAX\*EBX  
   add eax,2 ; EAX=EAX+2  
   xor edx,edx ; обнуляем EDX для корректной работы div  
   mov ebx,5 ; EBX=5  
   div ebx ; EAX=EAX/5, EDX=остаток от деления  
   mov edi,eax ; запись результата вычисления в 'edi'  
    
   ; ---- Вывод результата на экран  
   mov eax,div ; вызов подпрограммы печати  
   call sprint ; сообщения 'Результат: '  
   mov eax,edi ; вызов подпрограммы печати значения  
   call iprintLF ; из 'edi' в виде символов  
   mov eax,rem ; вызов подпрограммы печати  
   call sprint ; сообщения 'Остаток от деления: '  
   mov eax,edx ; вызов подпрограммы печати значения  
   call iprintLF ; из 'edx' (остаток) в виде символов  
    
   call quit ; вызов подпрограммы завершения

1. variant.asm

* ;--------------------------------  
   ; Программа вычисления варианта  
   ;--------------------------------  
   %include 'in\_out.asm'  
    
   SECTION .data  
   msg: DB 'Введите No студенческого билета: ',0  
   rem: DB 'Ваш вариант: ',0  
   SECTION .bss  
   x: RESB 80  
    
   SECTION .text  
   GLOBAL \_start  
   \_start:  
    
   mov eax, msg  
   call sprintLF  
    
   mov ecx, x  
   mov edx, 80  
   call sread  
    
   mov eax,x ; вызов подпрограммы преобразования  
   call atoi ; ASCII кода в число, `eax=x`  
    
   xor edx,edx  
   mov ebx,20  
   div ebx  
   inc edx  
    
   mov eax,rem  
   call sprint  
   mov eax,edx  
   call iprintLF  
    
   call quit

1. samrabota.asm - самостоятельная работа

* ;--------------------------------  
   ; Программа вычисления функции  
   ;--------------------------------  
   %include 'in\_out.asm'  
    
   SECTION .data  
   msg: DB 'Введите значение х: ',0  
   rem: DB 'Ваш результат: ',0  
   SECTION .bss  
   x: RESB 80  
    
   SECTION .text  
   GLOBAL \_start  
   \_start:  
    
   mov eax, msg  
   call sprintLF  
    
   mov ecx, x  
   mov edx, 80  
   call sread  
    
   mov eax,x ; вызов подпрограммы преобразования  
   call atoi ; ASCII кода в число, `eax=x`  
    
   add eax, 18  
   mov ebx,5  
   mul ebx  
    
   xor edx,edx  
   mov ebx, 28  
   neg ebx  
   add eax, ebx  
    
   mov edx, eax  
   mov eax,rem  
   call sprint  
   mov eax,edx  
   call iprintLF  
    
   call quit

# 6 Выводы

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

# Список литературы

1. [Текстовый документ “Лабораторная работа №7. Арифметические операции в NASM.”](https://esystem.rudn.ru/pluginfile.php/1584637/mod_resource/content/1/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%967.pdf)