

# **Лабораторная работа №6**

**Архитектура вычислительных систем**

Дадилов Руслан Магомедович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
<b>5</b>	<b>Ответы на вопросы:</b>	<b>19</b>
<b>6</b>	<b>Выводы</b>	<b>20</b>
	<b>Список литературы</b>	<b>21</b>

## Список иллюстраций

4.1	61.png . . . . .	9
4.2	62.png . . . . .	10
4.3	63.png . . . . .	11
4.4	64.png . . . . .	11
4.5	65.png . . . . .	12
4.6	66.png . . . . .	12
4.7	67.png . . . . .	13
4.8	68.png . . . . .	13
4.9	69.png . . . . .	14
4.10	610.png . . . . .	14
4.11	611.png . . . . .	15
4.12	612.png . . . . .	15
4.13	613.png . . . . .	16
4.14	614png . . . . .	16
4.15	615png . . . . .	17
4.16	616png . . . . .	17
4.17	617png . . . . .	18

## Список таблиц

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

## 2 Задание

Написать программу вычисления выражения. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения  $x$ , вычислять заданное выражение в зависимости от введенного  $x$ , выводить результат вычислений. Вид функции выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создать исполняемый файл и проверить его работу для значений из 6.3.

### 3 Теоретическое введение

1. Адресация в NASM 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.
2. Арифметические операции в NASM Схема команды целочисленного сложения `add` (от англ. addition - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда. Команда `add` работает как с числами со знаком, так и без знака.
3. Целочисленное вычитание `sub` Команда целочисленного вычитания `sub` (от англ. subtraction – вычитание) работает аналогично команде `add`.
4. Команды инкремента и декремента 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.
5. Команда изменения знака операнда `neg` Команда рассматривает свой операнд как число со знаком и меняет знак операнда на противоположный. Операндом может быть регистр или ячейка памяти любого размера.

6. Команды умножения `mul` и `imul` Умножение и деление, в отличие 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). Для знакового умножения используется команда `imul`.
7. Команды деления `div` и `idiv` Для деления, как и для умножения, существует 2 команды `div` (от англ. `divide` - деление) и `idiv`. Для беззнакового умножения используется команда `div`. Для знакового умножения используется команда `idiv`.



## 4 Выполнение лабораторной работы

1. Создадим директорию для лабораторной работы №6.
2. Перейдем в нее и создадим файл lab6-1.asm.


A terminal window screenshot with a light gray title bar. The title bar contains a square icon with a plus sign on the left and the text 'rmdadilov@dk5n59:~/work/arch-pc/lab06' on the right. The terminal shows four lines of commands and their outputs. The first line is 'mkdir ~/work/arch-pc/lab06'. The second line is 'cd ~/work/arch-pc/lab06'. The third line is 'touch lab6-1.asm'. The fourth line is 'nano lab6-1.asm'.

```
rmdadilov@dk5n59 ~/work/arch-pc $ mkdir ~/work/arch-pc/lab06
rmdadilov@dk5n59 ~/work/arch-pc $ cd ~/work/arch-pc/lab06
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ touch lab6-1.asm
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ nano lab6-1.asm
```

Рис. 4.1: 61.png

3. Введем в файл lab6-1.asm текст программы из листинга 6.1.

```
rmdadilov@dk3n31 ~/work/arch-pc/lab04 $ touch hello.asm
rmdadilov@dk3n31 ~/work/arch-pc/lab04 $ gedit hello.asm
```

Открыть ▾ 

hello.asm  
~/work/arch-pc/lab04

```
1 ; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Hello world!',10 ; 'Hello world!' плюс
4 ; символ перевода строки
5 helloLen: EQU $-hello ; Длина строки hello
6 SECTION .text ; Начало секции кода
7 GLOBAL _start
8 _start: ; Точка входа в программу
9 mov eax,4 ; Системный вызов для записи (sys_write)
10 mov ebx,1 ; Описатель файла '1' - стандартный вывод
11 mov ecx,hello ; Адрес строки hello в ecx
12 mov edx,helloLen ; Размер строки hello
13 int 80h ; Вызов ядра
14 mov eax,1 ; Системный вызов для выхода (sys_exit)
15 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
16 int 80h ; Вызов ядра
```

Рис. 4.2: 62.png

4. Создадим копию файла in\_out.asm в каталоге.

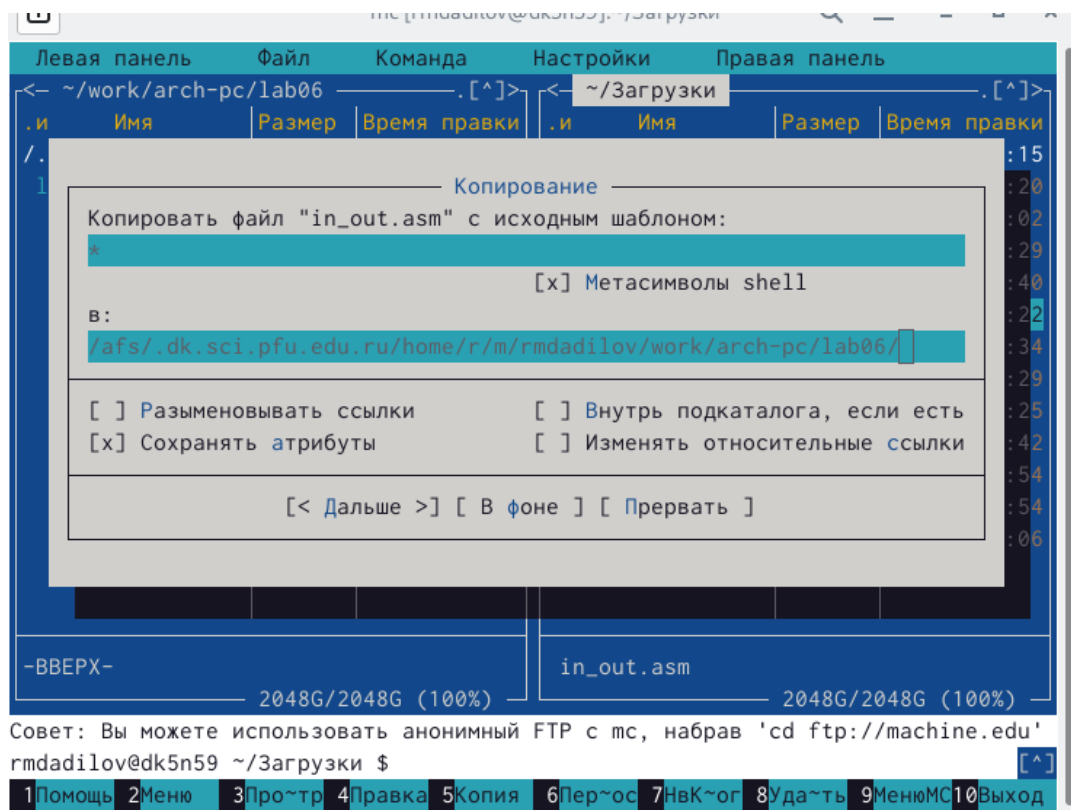


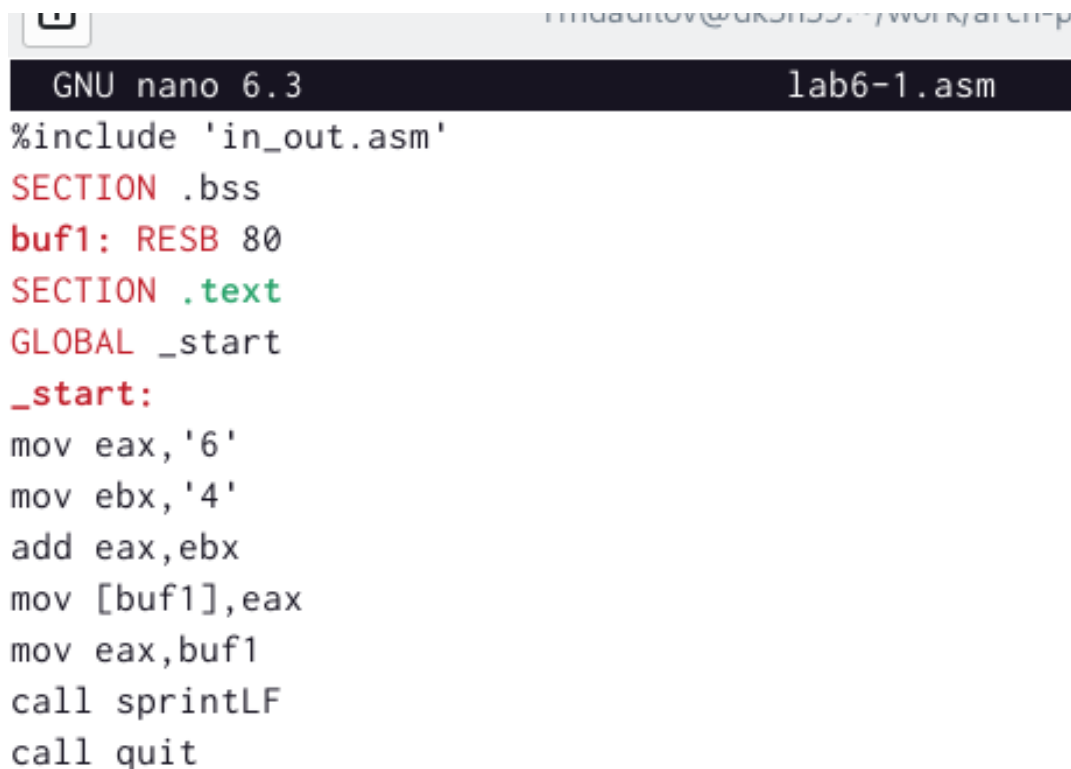
Рис. 4.3: 63.png

5. Создадим исполняемый файл и запустим его.

```
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ ./lab6-1
j
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $
```

Рис. 4.4: 64.png

6. Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправим текст программы.



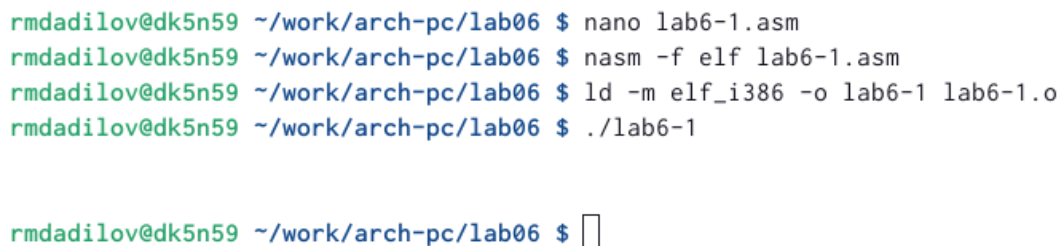
```

GNU nano 6.3                                lab6-1.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

```

Рис. 4.5: 65.png

7. Создадим исполняемый файл и запустим его (6-1).



```

rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ nano lab6-1.asm
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ ./lab6-1

rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ █

```

Рис. 4.6: 66.png



```
GNU nano 6.3 lab6-1.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
```

Рис. 4.7: 67.png

8. Создадим файл lab6-2.asm в каталоге. Введем в него текст программы из листинга 6.2 и запустим его.

```
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ touch lab6-2.asm
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ nano lab6-2.asm
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ ./lab6-2
106
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $
```

Рис. 4.8: 68.png

9. Изменим символы на числа в lab6-2. Создадим исполняемый файл и запустим его.



```
GNU nano 6.3 lab6-2.asm
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintLF
call quit
```

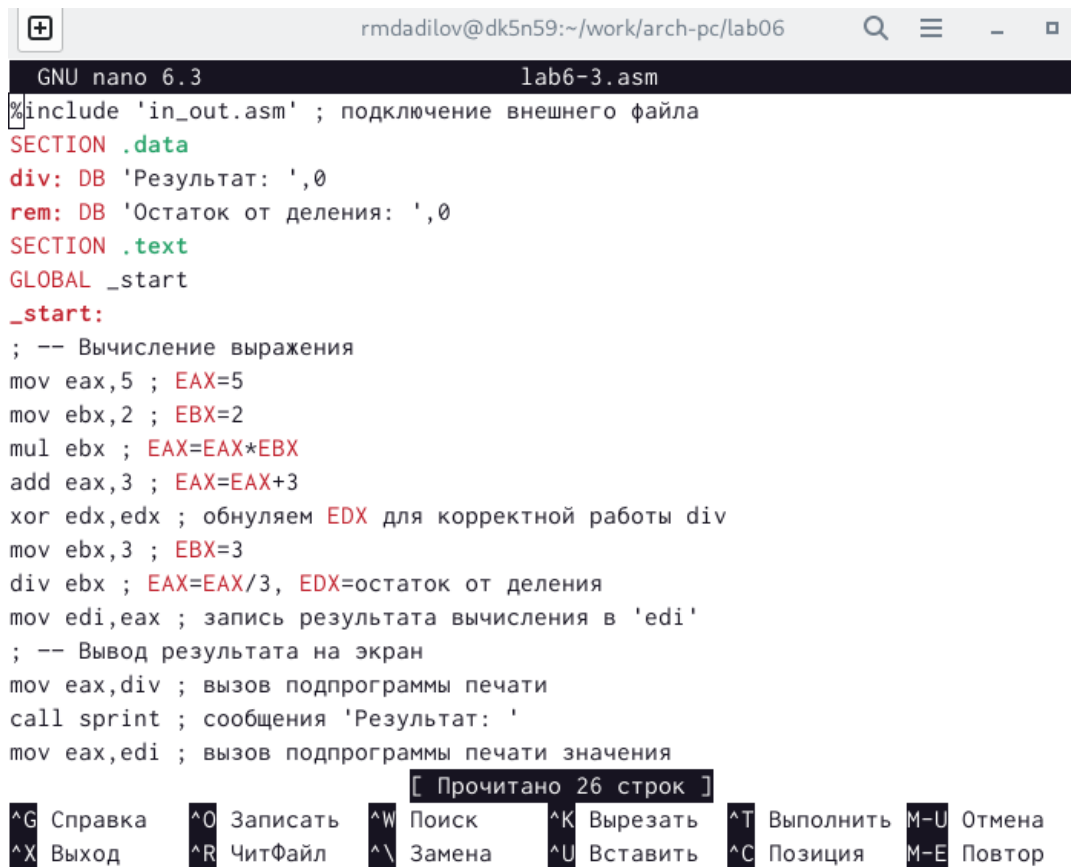
Рис. 4.9: 69.png

10. Создадим файл lab6-3.asm в каталоге. Введем в файл lab6-3.asm текст программы из листинга 6.3

```
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ nano lab6-2.asm
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ ./lab6-2
10
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $
```

Рис. 4.10: 610.png

11. Создадим исполняемый файл и запустим его.



```
GNU nano 6.3 lab6-3.asm
%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,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, 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
[ Прочитано 26 строк ]
^G Справка      ^O Записать     ^W Поиск        ^K Вырезать     ^T Выполнить   M-U Отмена
^X Выход        ^R ЧитФайл     ^\ Замена      ^U Вставить     ^C Позиция     M-E Повтор
```

Рис. 4.11: 611.png

12. Введем в файл lab6-3 программу вычисления выражения .

```
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ nano lab6-3.asm
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $
```

Рис. 4.12: 612.png

13. Создадим исполняемый файл и запустим его для вычисления выражения.

```
GNU nano 6.3 lab6-3.asm Изменён
#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,3 ; EBX=3
div ebx ; EAX=EAX/3, 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

^G Справка      ^O Записать     ^W Поиск        ^K Вырезать     ^T Выполнить    M-U Отмена
^X Выход        ^R ЧитФайл     ^\ Замена       ^U Вставить     ^C Позиция      M-E Повтор
```

Рис. 4.13: 613.png

14. Создадим файл variant.asm в каталоге ~/work/arch-pc/lab06. После в файл вводим номер студенческого и получаем вариант для выполнения задания

```
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ nano lab6-3.asm
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $
```

Рис. 4.14: 614png



```

rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ touch variant.asm
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ nano variant.asm
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132227120
Ваш вариант: 1

```

Рис. 4.15: 615png

16. Составляем программу для нашего варианта lab6-4 (Самостоятельная работа).

```

GNU nano 6.3 lab6-4.asm
mov eax,rem
call sprintLF
mov eax, rem1
call sprint
mov ecx, x
mov edx,80
call sread
mov eax, x
call atoi
mov ebx,10
mul ebx
add eax,2
xor edx,edx
mov ebx,3
div ebx
mov edi,eax
mov eax, div
call sprint
mov eax, edi
call iprintLF

```

<sup>^</sup>G Справка    <sup>^</sup>O Записать    <sup>^</sup>W Поиск    <sup>^</sup>K Вырезать    <sup>^</sup>T Выполнить    M-U Отм  
<sup>^</sup>X Выход    <sup>^</sup>R ЧитФайл    <sup>^</sup>\ Замена    <sup>^</sup>U Вставить    <sup>^</sup>C Позиция    M-E Пов

Рис. 4.16: 616png

17. Запускаем программу и вводим два числа из условия, убеждаемся что программа работает верно (Самостоятельная работа).

```
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ ./lab6-4
Введите переменную x:
x будет 1
Результат :4
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ ./lab6-4
Введите переменную x:
x будет 10
Результат :34
rmdadilov@dk5n59 ~/work/arch-pc/lab06 $ █
```

Рис. 4.17: 617png

## 5 Ответы на вопросы:

1. `mov eax` и `rem call sprint` отвечают за вывод на экран сообщения 'Ваш вариант:';
2. `mov ecx,x` - запись входной переменной в регистр `ecx`; `mov edx, 80` - запись размера переменной в регистр `edx`; `call sread` - вызов процедуры чтения данных;
3. `call atoi` - функция преобразующая ASCII код символа в целое число и записывающая результат в регистр `eax`;
4. `xor edx, edx` `mov ebx, 20` `div ebx`, `inc edx`;
5. `div ebx` - `ebx` записывается остаток от деления при выполнении инструкции "div ebx";
6. "inc edx" - используется для увеличения операнда на единицу;
7. `mov eax, rem` `call sprint` `mov eax, edx` `call iprintLF` строки листинга 7.4 отвечают за вывод на экран результата вычислений.

## 6 Выводы

В ходе выполнения данной лабораторной работы были освоены арифметические инструкции языка ассемблера NASM.

## **Список литературы**