

# **Лаборная работа №7 Арифметические операции в NASM**

**НММ-6д-02-22**

Крухмалев Артём Владиславович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>3</b>
<b>2</b>	<b>Задание</b>	<b>4</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>5</b>
<b>4</b>	<b>Самостоятельная работа</b>	<b>11</b>
<b>5</b>	<b>Выводы</b>	<b>12</b>

# **1 Цель работы**

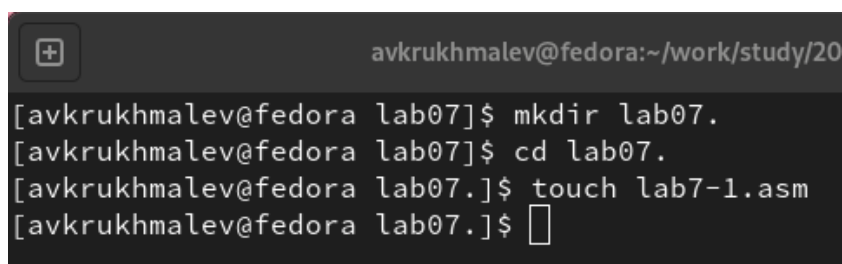
Научиться работать с арифметическими операциями

## 2 Задание

Написать программу для вычисления предложенной функции

### 3 Выполнение лабораторной работы

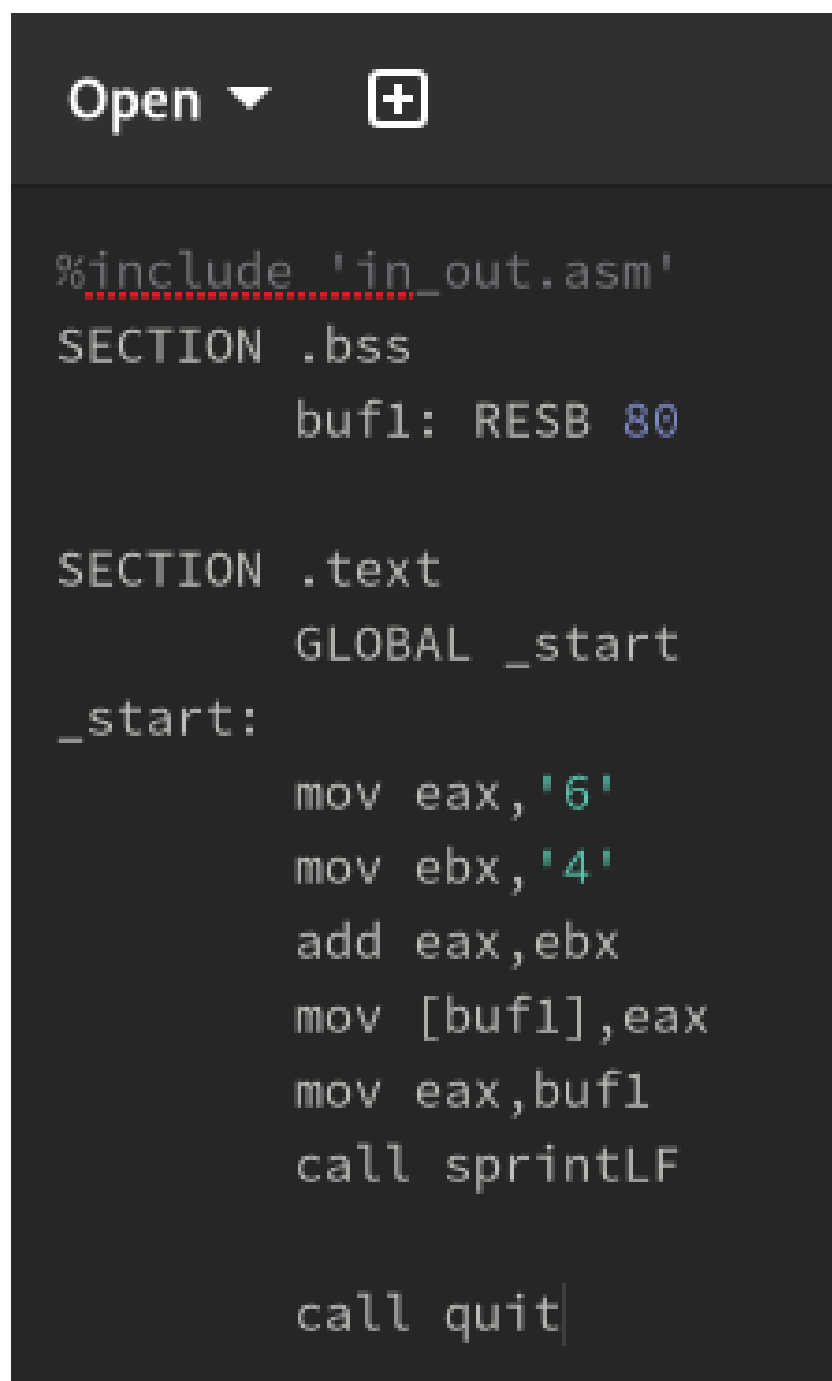
1. С помощью терминала создадим подкаталог

A terminal window with a dark background. The title bar shows a window icon and the text 'avkrukhmalev@fedora:~/work/study/20'. The terminal content shows four lines of commands and their outputs: '[avkrukhmalev@fedora lab07]\$ mkdir lab07.', '[avkrukhmalev@fedora lab07]\$ cd lab07.', '[avkrukhmalev@fedora lab07.]\$ touch lab7-1.asm', and '[avkrukhmalev@fedora lab07.]\$' followed by a cursor.

```
avkrukhmalev@fedora:~/work/study/20
[avkrukhmalev@fedora lab07]$ mkdir lab07.
[avkrukhmalev@fedora lab07]$ cd lab07.
[avkrukhmalev@fedora lab07.]$ touch lab7-1.asm
[avkrukhmalev@fedora lab07.]$
```

Рис. 3.1: Новый каталог

2. Напишем программу для вычисления суммы заданных переменных

The image shows a code editor window with a dark background. At the top, there is a header bar with the word "Open" followed by a downward-pointing triangle and a square icon containing a plus sign. The main area of the editor contains assembly code. The code starts with an include directive for 'in\_out.asm', followed by a section declaration for '.bss' containing a buffer 'buf1' of size 80. Then, a section declaration for '.text' is shown, along with a global symbol '\_start'. The '\_start' label marks the beginning of the program, where the value 6 is moved into the 'eax' register, the value 4 is moved into the 'ebx' register, they are added together, the result is stored in 'buf1', and then 'buf1' is moved back into 'eax'. Finally, the 'sprintLF' function is called to print the result, followed by a 'quit' function call.


```
Open ▼   
  
%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. 3.2: Сумма

### 3. Создадим файл программы

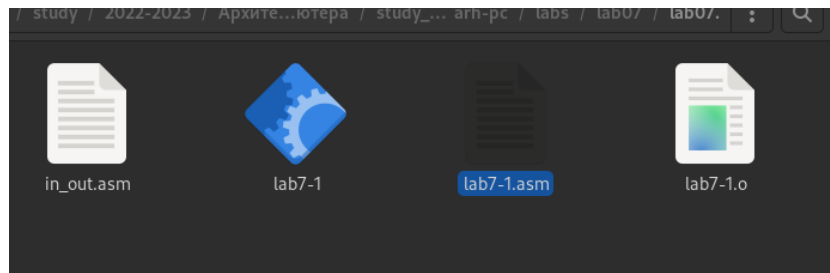


Рис. 3.3: Откомпилированный файл

4. Запустим программу, заметим, что вывелось не число, а буква j

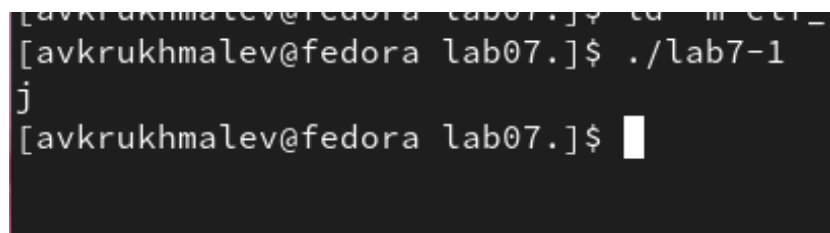


Рис. 3.4: Вывод переменной

5. Изменим программу и посмотрим, что изменится, заметим, что символ соответствующий 10 не выводится терминалом. 10 соответствует символы LF, /n

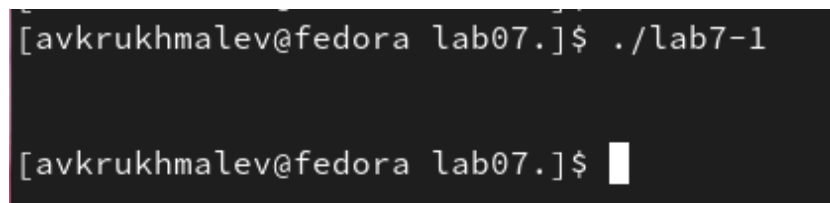


Рис. 3.5: Вывод после изменений

6. Создадим следующий файл, скомпилируем его и посмотрим, что он выведет-106, сумма кодов символов

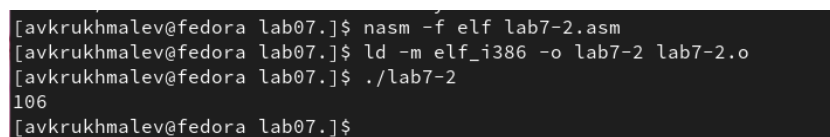
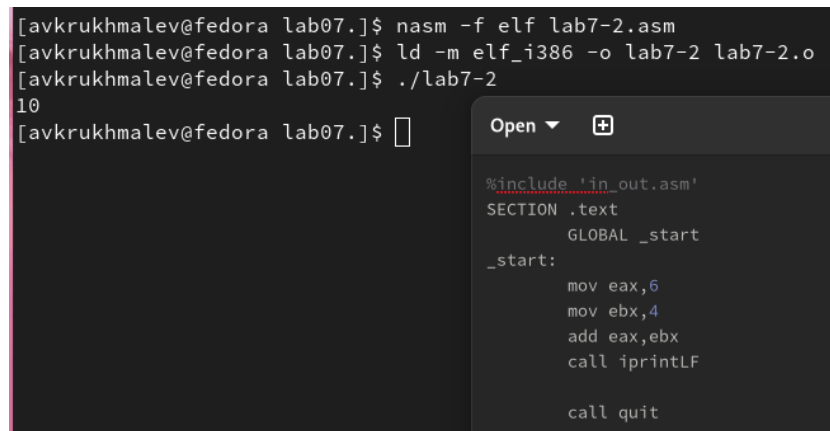


Рис. 3.6: Вывод 2й программы

7. Изменим программу, выведем ответ с помощью `iprintLF` и `iprint`. Отличие в выводе на следующей или на этой же строке

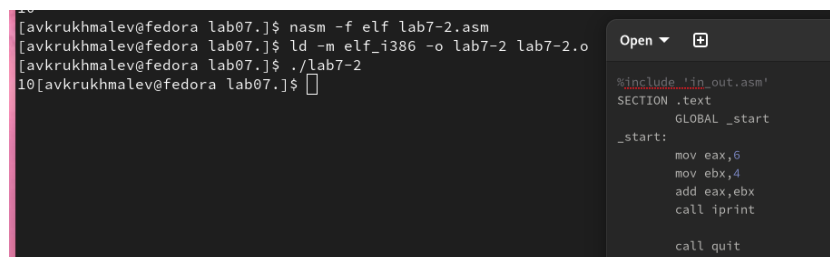


```
[avkrukhmalev@fedora lab07.]$ nasm -f elf lab7-2.asm
[avkrukhmalev@fedora lab07.]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[avkrukhmalev@fedora lab07.]$ ./lab7-2
10
[avkrukhmalev@fedora lab07.]$
```

```
%include 'in_out.asm'
SECTION .text
    GLOBAL _start
_start:
    mov eax,6
    mov ebx,4
    add eax,ebx
    call iprintLF

    call quit
```

Рис. 3.7: `iprintLF`



```
10
[avkrukhmalev@fedora lab07.]$ nasm -f elf lab7-2.asm
[avkrukhmalev@fedora lab07.]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[avkrukhmalev@fedora lab07.]$ ./lab7-2
10[avkrukhmalev@fedora lab07.]$
```

```
%include 'in_out.asm'
SECTION .text
    GLOBAL _start
_start:
    mov eax,6
    mov ebx,4
    add eax,ebx
    call iprint

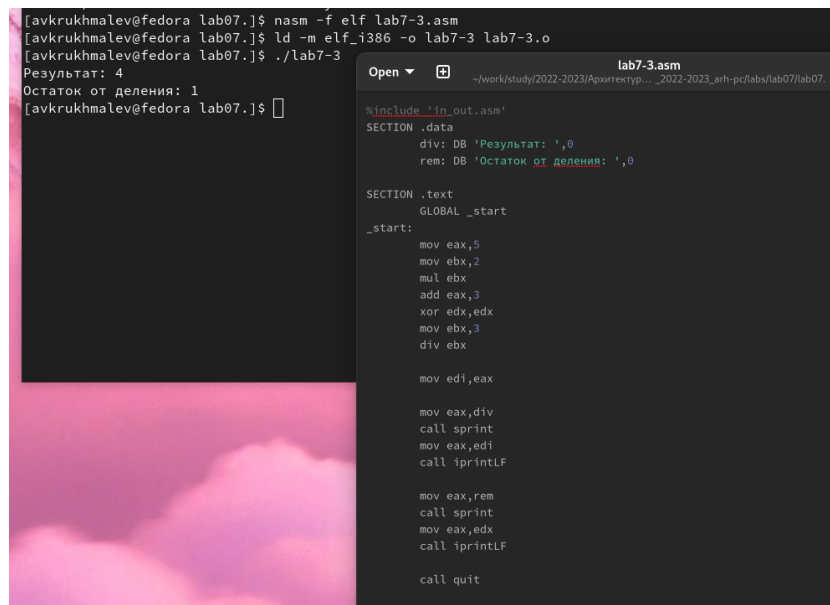
    call quit
```

Рис. 3.8: `iprint`

8. Создадим следующий файл по вычислению обычной функции без неизвестных.



```
[avkrukhmalev@fedora lab07.]$ nasm -f elf lab7-3.asm
[avkrukhmalev@fedora lab07.]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[avkrukhmalev@fedora lab07.]$ ./lab7-3
Результат: 4
Остаток от деления: 1
[avkrukhmalev@fedora lab07.]$
```



```
%include 'in_out.asm'
SECTION .data
    div: DB 'Результат: ',0
    rem: DB 'Остаток от деления: ',0

SECTION .text
    GLOBAL _start
_start:
    mov eax,5
    mov ebx,2
    mul ebx
    add eax,3
    xor edx,edx
    mov ebx,3
    div ebx

    mov edi,eax

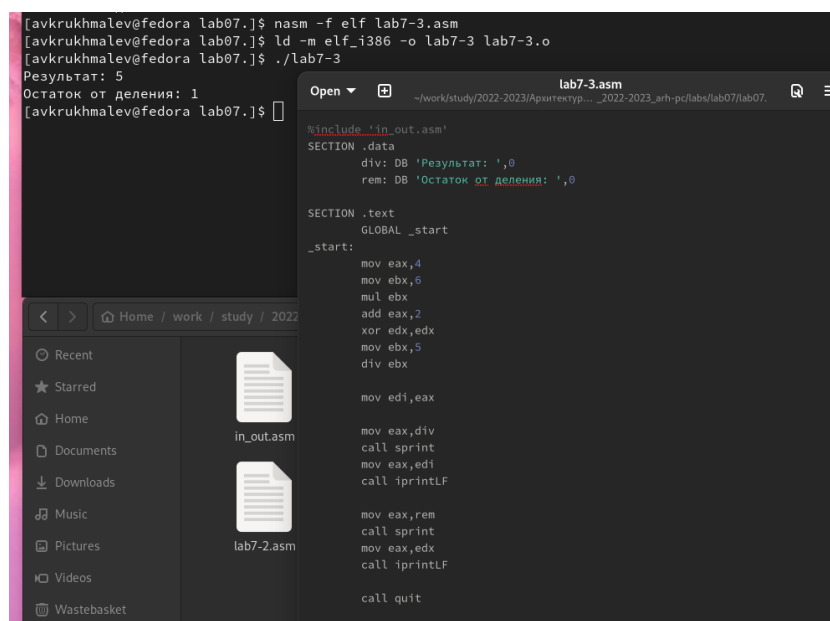
    mov eax,div
    call sprint
    mov eax,edi
    call iprintLF

    mov eax,rem
    call sprint
    mov eax,edx
    call iprintLF

    call quit
```

Рис. 3.9: простейшая функция

```
[avkrukhmalev@fedora lab07.]$ nasm -f elf lab7-3.asm
[avkrukhmalev@fedora lab07.]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[avkrukhmalev@fedora lab07.]$ ./lab7-3
Результат: 5
Остаток от деления: 1
[avkrukhmalev@fedora lab07.]$
```



```
%include 'in_out.asm'
SECTION .data
    div: DB 'Результат: ',0
    rem: DB 'Остаток от деления: ',0

SECTION .text
    GLOBAL _start
_start:
    mov eax,4
    mov ebx,6
    mul ebx
    add eax,2
    xor edx,edx
    mov ebx,5
    div ebx

    mov edi,eax

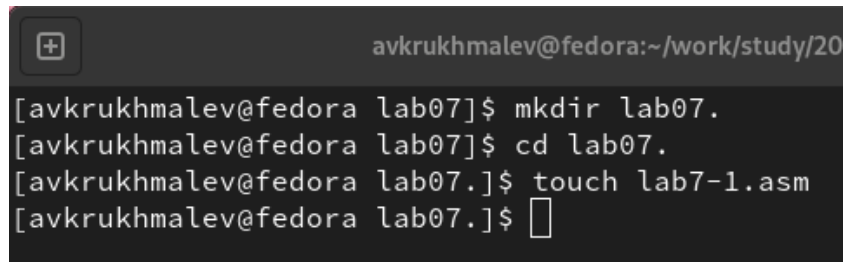
    mov eax,div
    call sprint
    mov eax,edi
    call iprintLF

    mov eax,rem
    call sprint
    mov eax,edx
    call iprintLF

    call quit
```

Рис. 3.10: проверка с другими числами

10. Создадим новый файл и узнаем вариант самостоятельной работы



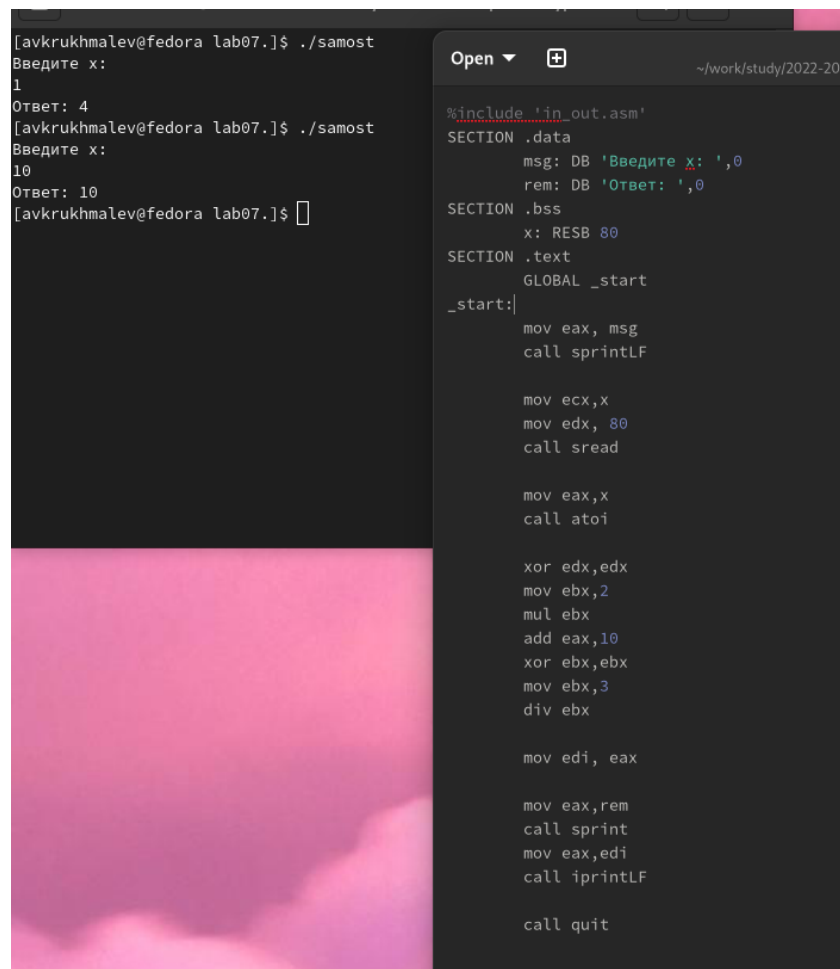
```
avkrukhmalev@fedora:~/work/study/20
[avkrukhmalev@fedora lab07]$ mkdir lab07.
[avkrukhmalev@fedora lab07]$ cd lab07.
[avkrukhmalev@fedora lab07.]$ touch lab7-1.asm
[avkrukhmalev@fedora lab07.]$
```

Рис. 3.11: 1 Вариант

11. 1.С помощью следующих строк выводится надпись “Ваш вариант” `mov eax,rem call sprint mov eax,edx call iprintLF` 2.Для записи x с клавиатуры и вывода сообщения для этого 3.Для того чтобы преобразовать код символа в нужное нам число 4.Данные строки отвечают за просчет варианта `xor edx,edx mov ebx,20 div ebx inc edx` 5.В `edx` 6.`inc` прибавляет единицу 7.Данные строки отвечают за вывод текста `mov eax,rem call sprint mov eax,edx call iprintLF`

## 4 Самостоятельная работа

1. Мне попался 1 вариант, напишем код и выведем результат



The image shows a terminal window on the left and a code editor on the right. The terminal displays the execution of a program named `./samost`. It prompts the user to enter a value `x`. In the first run, the user enters `1`, and the program outputs `Ответ: 4`. In the second run, the user enters `10`, and the program outputs `Ответ: 10`. The code editor on the right shows the assembly code for the program. It includes a header file `'in_out.asm'` and defines a data section with a message `msg` and a remainder `rem`. The .bss section reserves 80 bytes for `x`. The .text section contains the main logic: it prints the message, reads 80 bytes into `x`, converts it to an integer, calculates `(x * 2 + 10) / 3`, and prints the result.

```
[avkrukhmalev@fedora lab07.]$ ./samost
Введите x:
1
Ответ: 4
[avkrukhmalev@fedora lab07.]$ ./samost
Введите x:
10
Ответ: 10
[avkrukhmalev@fedora lab07.]$
```

```
%include 'in_out.asm'
SECTION .data
    msg: DB 'Введите x: ',0
    rem: DB 'Ответ: ',0
SECTION .bss
    x: RESB 80
SECTION .text
    GLOBAL _start
_start:
    mov eax, msg
    call sprintf

    mov ecx, x
    mov edx, 80
    call sread

    mov eax, x
    call atoi

    xor edx, edx
    mov ebx, 2
    mul ebx
    add eax, 10
    xor ebx, ebx
    mov ebx, 3
    div ebx

    mov edi, eax

    mov eax, rem
    call sprintf
    mov eax, edi
    call iprintLF

    call quit
```

Рис. 4.1: Вывод

## 5 Выводы

В данной работе мы познакомились с операторами для вычислений, а также написали программу для вычисления с неизвестной