

# **Лабораторная работа №6**

**Архитектура вычислительных систем**

Кавказова Диана Алексеевна

# 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 . . . . .	13
4.7	67.png . . . . .	13
4.8	68.png . . . . .	14
4.9	69.png . . . . .	14
4.10	610.png . . . . .	15
4.11	611.png . . . . .	15
4.12	612.png . . . . .	16
4.13	613.png . . . . .	16
4.14	614png . . . . .	17
4.15	615png . . . . .	17
4.16	616png . . . . .	18
4.17	617png . . . . .	18

## **Список таблиц**

# 1 Цель работы

Освоить арифметические инструкции 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

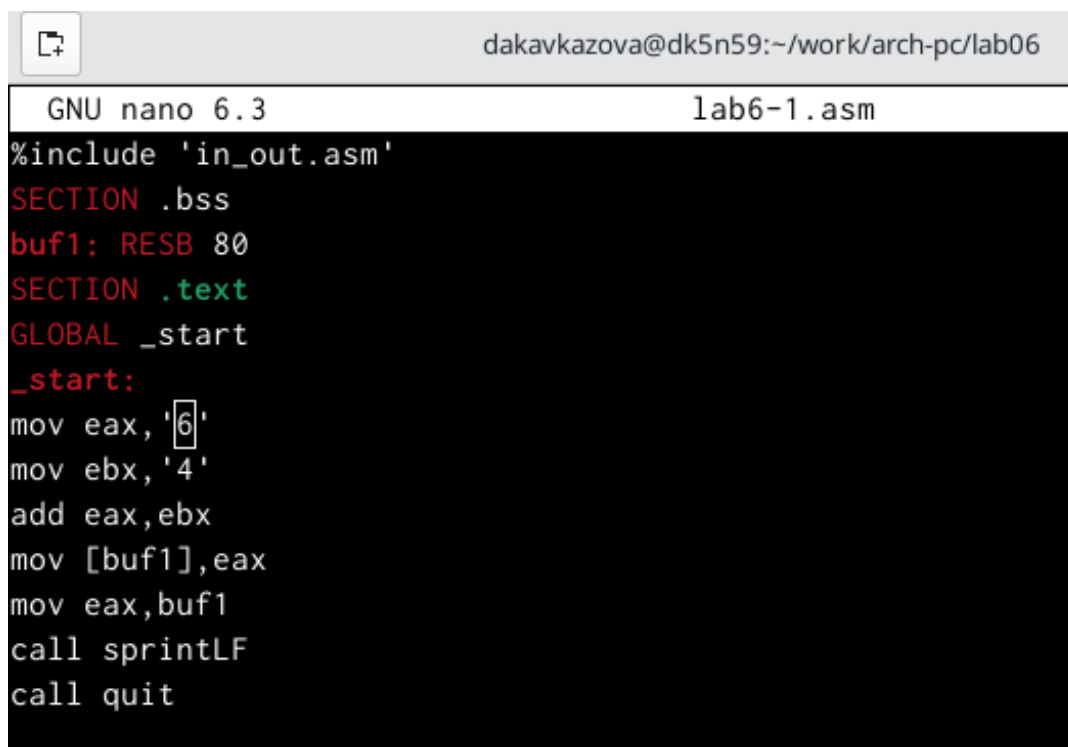
1. Создаём каталог для программ лабораторной работы No 7, перейдём в него и создаём файл lab6-1.asm



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

Рис. 4.1: 61.png

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



```
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. 4.2: 62.png

3. Создаём копию файла in\_out.asm в каталоге.

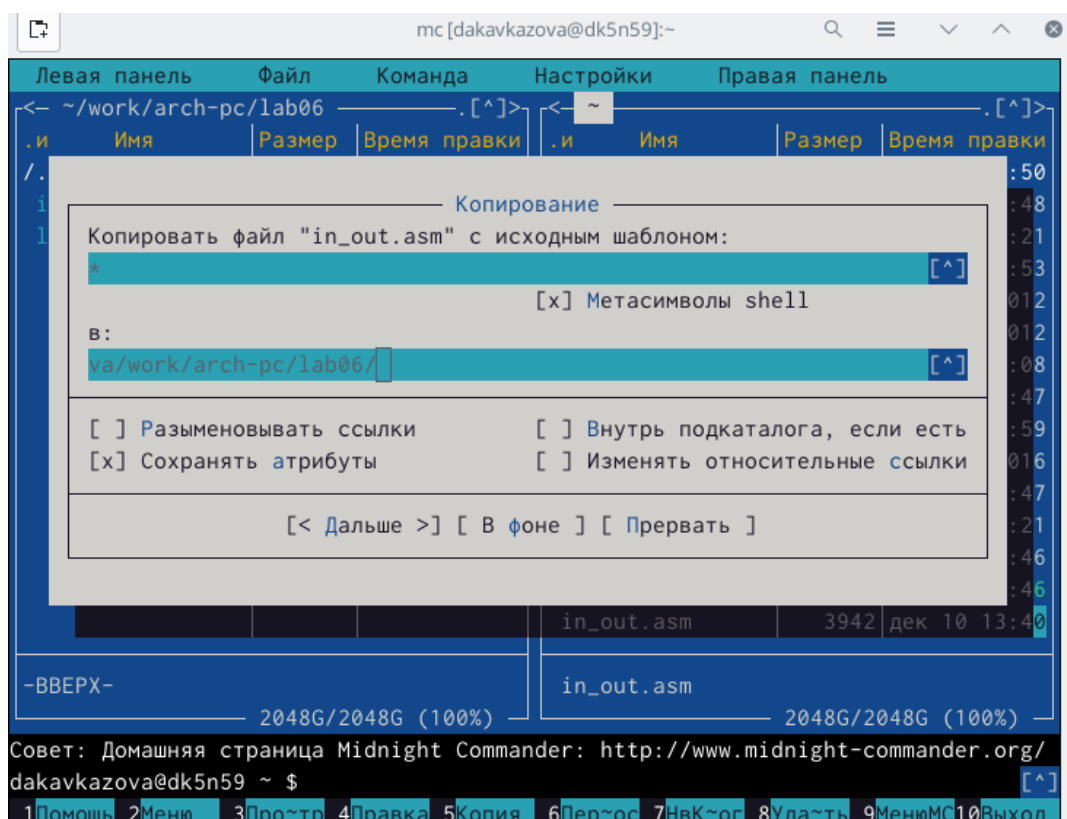


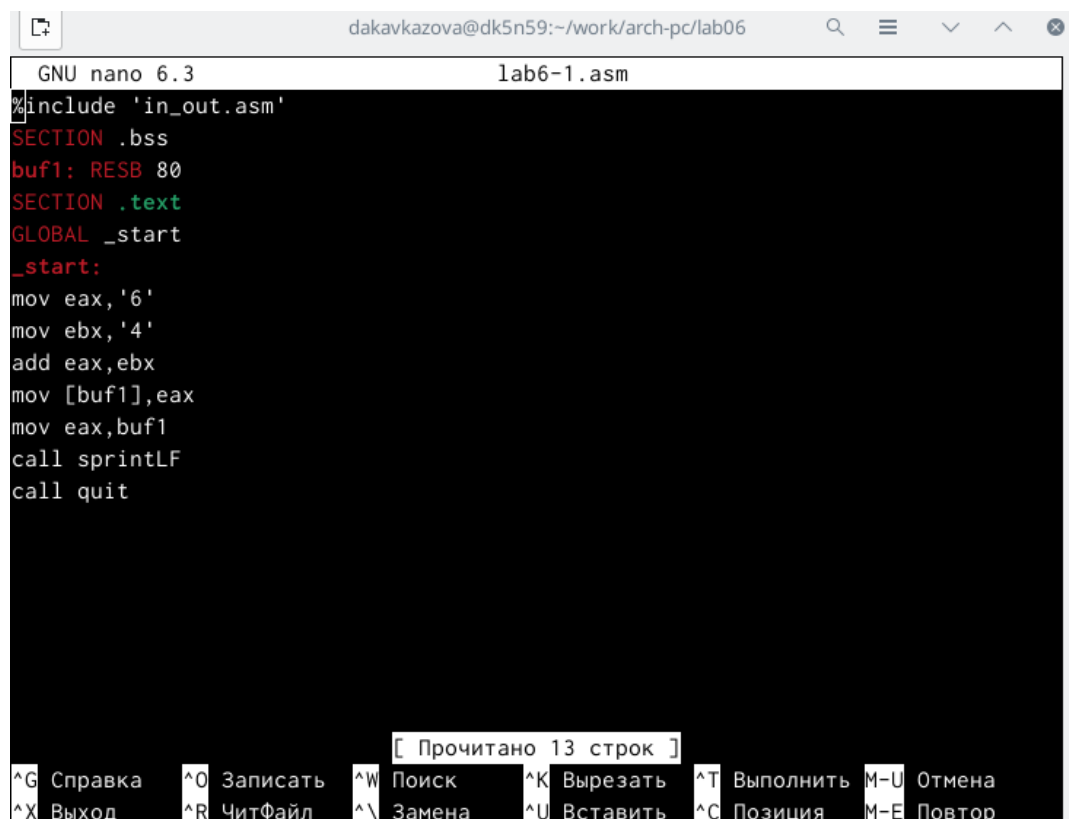
Рис. 4.3: 63.png

4. Создадим исполняемый файл и запустим его.

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

Рис. 4.4: 64.png

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



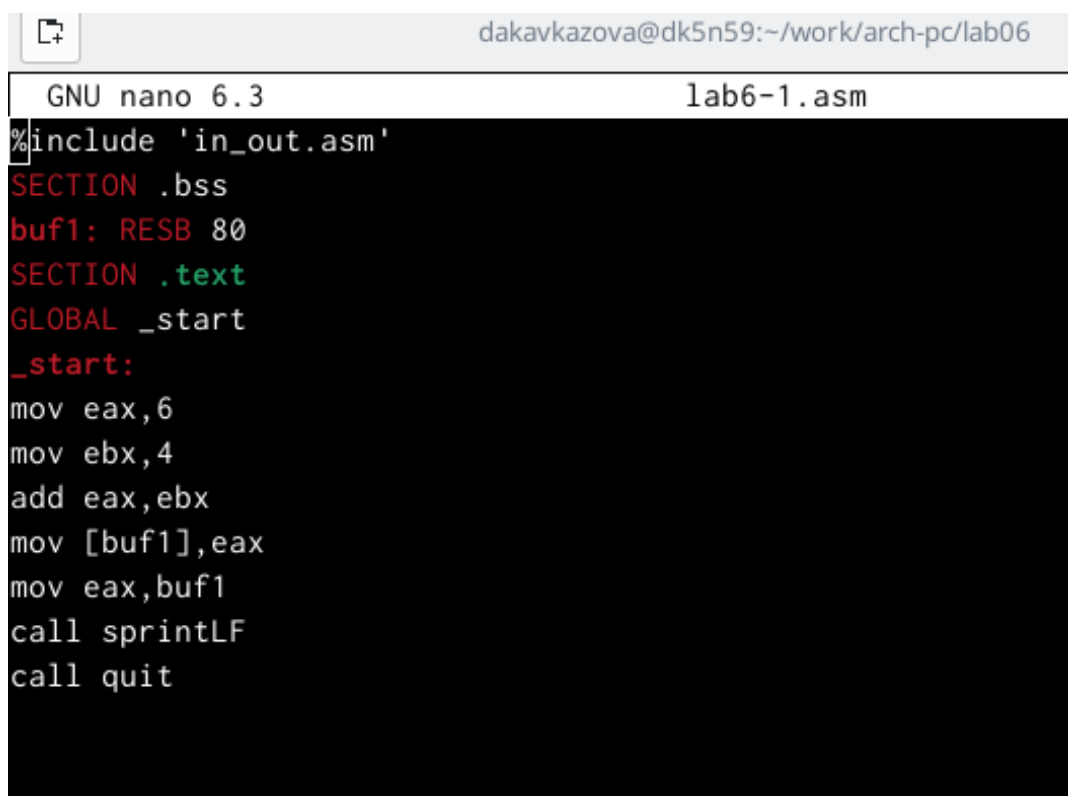
```
GNU nano 6.3 lab6-1.asm
%include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
mov [buf1], eax
mov eax, buf1
call sprintLF
call quit
```

[ Прочитано 13 строк ]

<b>^G</b> Справка	<b>^O</b> Записать	<b>^W</b> Поиск	<b>^K</b> Вырезать	<b>^T</b> Выполнить	<b>M-U</b> Отмена
<b>^X</b> Выход	<b>^R</b> ЧитФайл	<b>^N</b> Замена	<b>^U</b> Вставить	<b>^C</b> Позиция	<b>M-E</b> Повтор

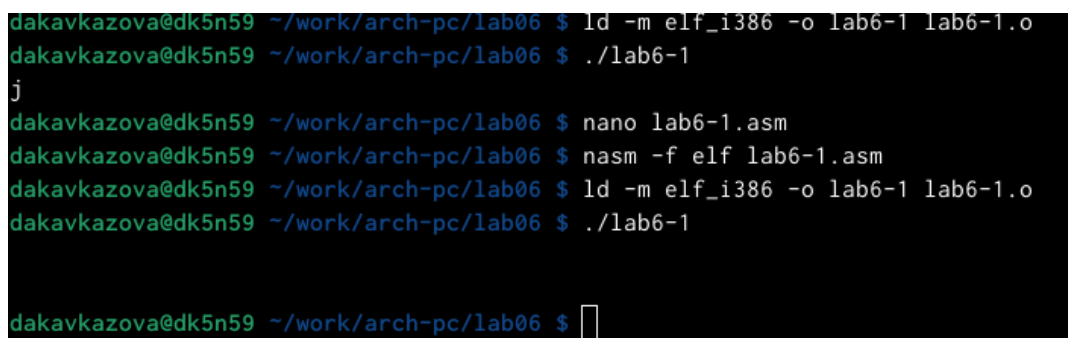
Рис. 4.5: 65.png

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



```
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. 4.6: 66.png



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

dakavkazova@dk5n59 ~/work/arch-pc/lab06 $
```

Рис. 4.7: 67.png

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

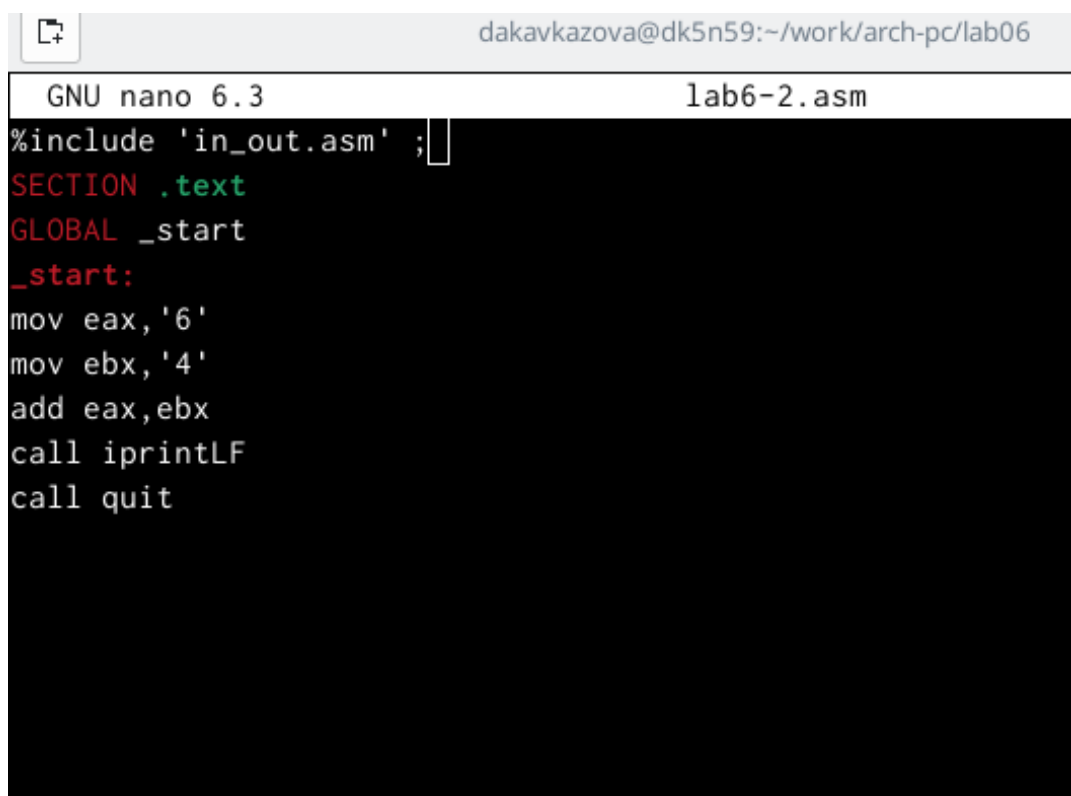
```

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

```

Рис. 4.8: 68.png

8. 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

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

```

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

```

Рис. 4.10: 610.png

10. Создадим исполняемый файл и запустим его.

```

GNU nano 6.3 variant.asm Изменён
%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, %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

```

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

Рис. 4.11: 611.png

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

```

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

```

Рис. 4.12: 612.png

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

```

dakavkazova@dk5n59 ~/work/arch-pc/lab06 $ gedit lab6-3.asm
dakavkazova@dk5n59 ~/work/arch-pc/lab06 $
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; -- Вычисление выражения
9 mov eax,4 ;
10 mov ebx,6 ;
11 mul ebx ;
12 add eax,2 ;
13 xor edx,edx ;
14 mov ebx,5 ;
15 div ebx ;
16 mov edi,eax ;
17 ; -- Вывод результата на экран
18 mov eax,div ;
19 call sprint ;
20 mov eax,edi ;
21 call iprintLF ;
22 mov eax,rem ;
23 call sprint ;

```

Рис. 4.13: 613.png

13. Создадим файл variant.asm в каталоге ~/work/arch-pc/lab06:



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

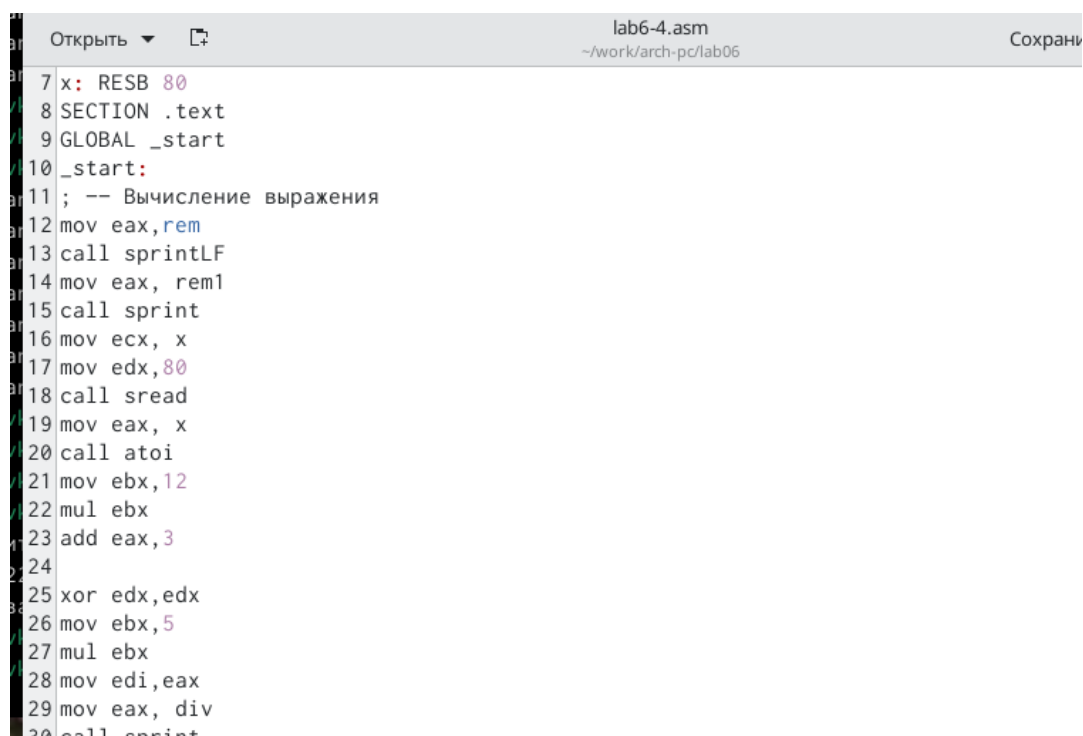
Рис. 4.14: 614png


14. Вводим номер студенческого и получаем вариант для выполнения задания

```
dakavkazova@dk5n59 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132222001
Ваш вариант: 2
dakavkazova@dk5n59 ~/work/arch-pc/lab06 $
```

Рис. 4.15: 615png

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

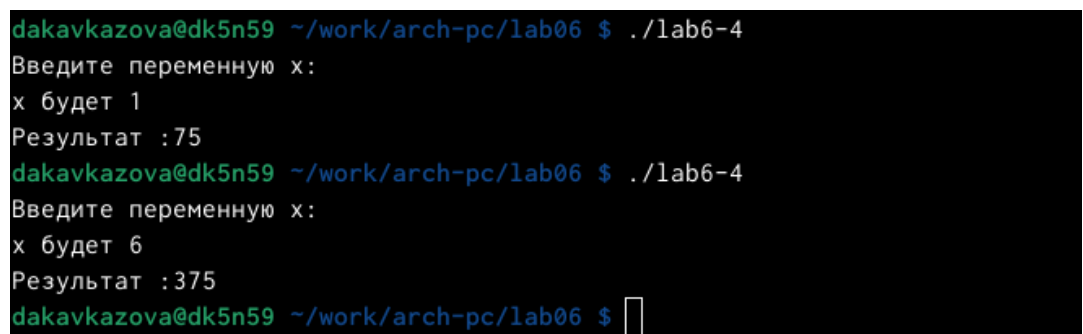


```
Открыть ▾  lab6-4.asm Сохрани
~/work/arch-pc/lab06

7 x: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11 ; -- Вычисление выражения
12 mov eax, rem
13 call sprintLF
14 mov eax, rem1
15 call sprint
16 mov ecx, x
17 mov edx, 80
18 call sread
19 mov eax, x
20 call atoi
21 mov ebx, 12
22 mul ebx
23 add eax, 3
24
25 xor edx, edx
26 mov ebx, 5
27 mul ebx
28 mov edi, eax
29 mov eax, div
30 call sprint
```

Рис. 4.16: 616png

16. Запускаем программу и вводим два числа из условия, убеждаемся что программа работает верно.



```
dakavkazova@dk5n59 ~/work/arch-pc/lab06 $ ./lab6-4
Введите переменную x:
x будет 1
Результат :75
dakavkazova@dk5n59 ~/work/arch-pc/lab06 $ ./lab6-4
Введите переменную x:
x будет 6
Результат :375
dakavkazova@dk5n59 ~/work/arch-pc/lab06 $ █
```

Рис. 4.17: 617png

## 5 Ответы на вопросы:

1. строки листинга 7.4 отвечают за вывод на экран сообщения 'Ваш вариант:  
mov eax и rem call sprint;
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 запи-  
сывающая результат в регистр eax;
4. xor edx, edx mov ebx, 20 div ebx, inc edx;
5. div ebx - ebx;
6. inc - используется для увеличения операнда на единицу;
7. Следующие строки листинга отвечают за вывод на экран результата вычис-  
лений mov eax, rem call sprint mov eax, edx call iprintLF.

## 6 Выводы

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

## **Список литературы**