

# Отчёт по лабораторной работе №6

## Дисциплина: архитектура компьютеров

Зайцева Ульяна Владимировна

### Содержание

1	Цель работы .....	1
2	Задание .....	1
3	Теоретическое введение.....	1
4	Выполнение лабораторной работы.....	2
5	Выводы.....	13

### 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

### 2 Задание

1. Символьные и численные данные в NASM.
2. Выполнение арифметических операций в NASM.
3. Ответы на вопросы по листингу 6.4
4. Задание для самостоятельной работы.

### 3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. 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.

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

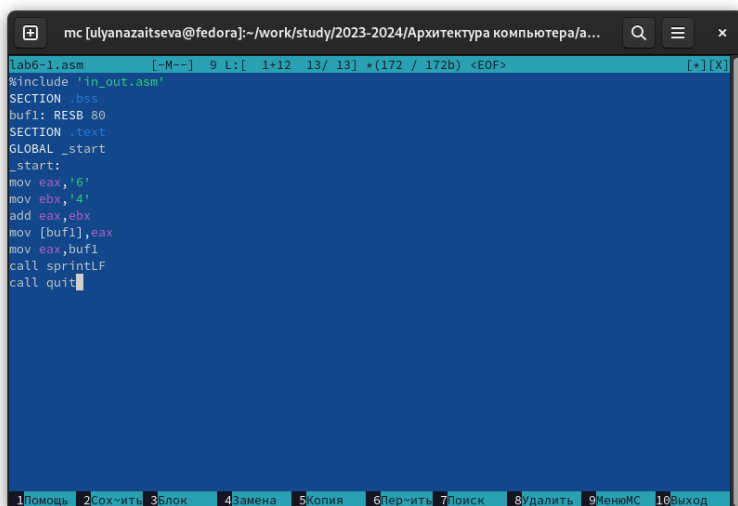
Команда целочисленного вычитания sub (от англ. subtraction – вычитание) работает аналогично команде add

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, а вторая (коды 128-255) предназначена для специальных символов и букв национальных алфавитов и на компьютерах разных типов может меняться.

## 4 Выполнение лабораторной работы

### 1. Символьные и численные данные в NASM.

Ввожу в файл lab6-1.asm текст программы из листинга 6.1.(рис. ??)



```
lab6-1.asm  [-M--]  9  L: [ 1+12 13/ 13] *(172 / 172b) <EOF>  [*] [X]
#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 sprintf
call quit
```

*Первый листинг*

Создаю исполняемый файл и запускаю его.(рис. ??)

```
ulyanazaitseva@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-p...
[ulyanazaitseva@fedora ~]$ cd ~/work/study/2023-2024/'Архитектура компьютера'/arch-pc/lab06
[ulyanazaitseva@fedora lab06]$ touch lab6-1.asm
[ulyanazaitseva@fedora lab06]$ mc

[ulyanazaitseva@fedora lab06]$ nasm -f elf lab6-1.asm
[ulyanazaitseva@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[ulyanazaitseva@fedora lab06]$ ./lab6-1
j
[ulyanazaitseva@fedora lab06]$
```

### *Запуск программы*

Далее меняю текст программы и вместо символов запишу числа. Исправляю текст программы:

Меняю строки

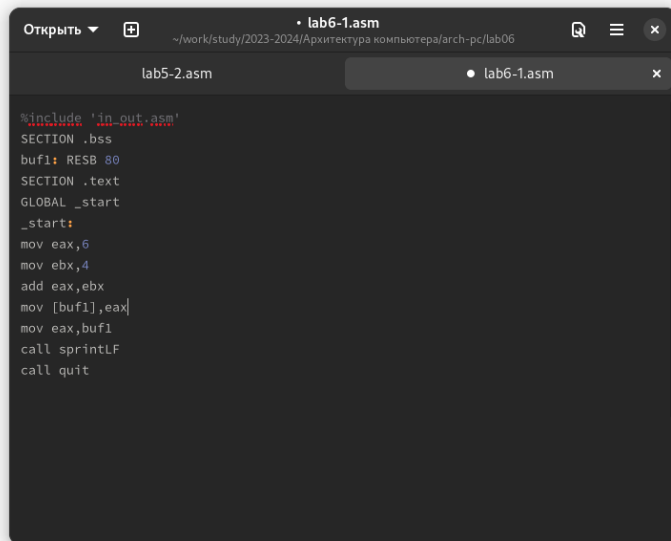
mov eax,'6'

mov ebx,'4'

на строки

mov eax,6

mov ebx,4 (рис. ??)



```
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 sprintf
call quit
```

### *Изменённая программа*

Создаю исполняемый файл и запускаю его.(рис. ??)

```
[ulyanazaitseva@fedora lab06]$ nasm -f elf lab6-1.asm
[ulyanazaitseva@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[ulyanazaitseva@fedora lab06]$ ./lab6-1

[ulyanazaitseva@fedora lab06]$
```

### *Выполнение программы*

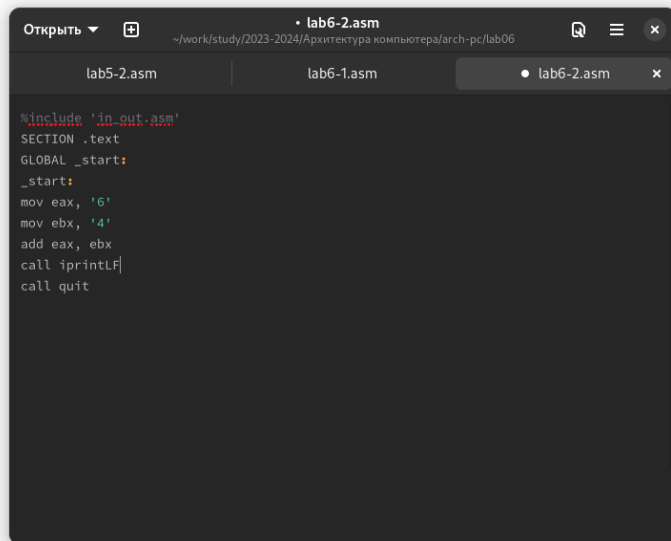
Этому коду (10) соответствует символ 'LF', который перемещает курсор на следующую строку. Символ при выводе не отображается.

Создаю файл lab6-2.asm в нужном каталоге(рис. ??)

```
[ulyanazaitseva@fedora lab06]$ touch lab6-2.asm
[ulyanazaitseva@fedora lab06]$
```

### *Выполнение программы*

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



```
%include 'in-out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call printf
call quit
```

### *Второй листинг*

Создаю исполняемый файл и запускаю его.(рис. ??)



```
[ulyanazaitseva@fedora lab06]$ nasm -f elf lab6-2.asm
[ulyanazaitseva@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[ulyanazaitseva@fedora lab06]$ ./lab6-2
106
[ulyanazaitseva@fedora lab06]$
```

### *Вывод программы*

В этой программе также заменяю строки

mov eax,'6'

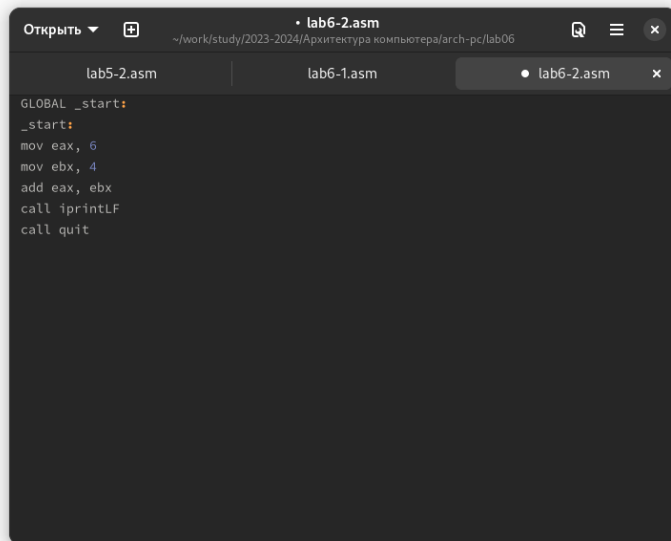
mov ebx,'4'

на строки

mov eax,6

mov ebx,4

(рис. ??).



*Изменяю программу*

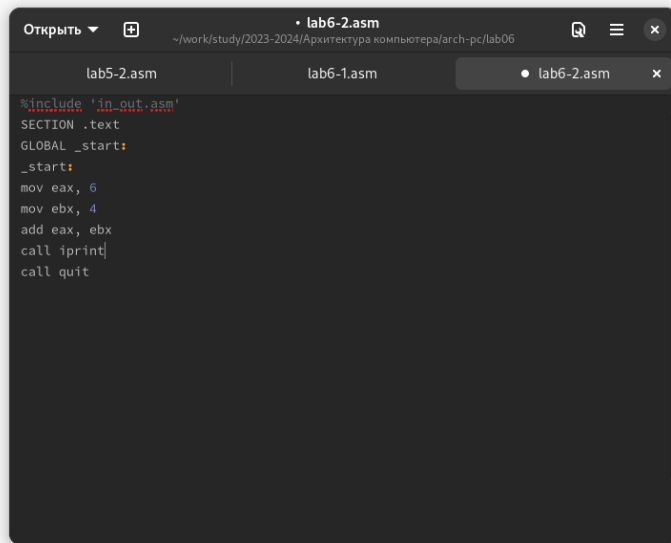
Создаю исполняемый файл и запускаю его.(рис. ??)

```
[ulyanazaitseva@fedora lab06]$ nasm -f elf lab6-2.asm
[ulyanazaitseva@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[ulyanazaitseva@fedora lab06]$ ./lab6-2
10
[ulyanazaitseva@fedora lab06]$
```

*Проверка работы программы*

В выводе получаю число 10.

Заменяю функцию iprintLF на iprint.(рис. ??)



### *Замена в программе*

Создаю исполняемый файл и запускаю его.(рис. ??)

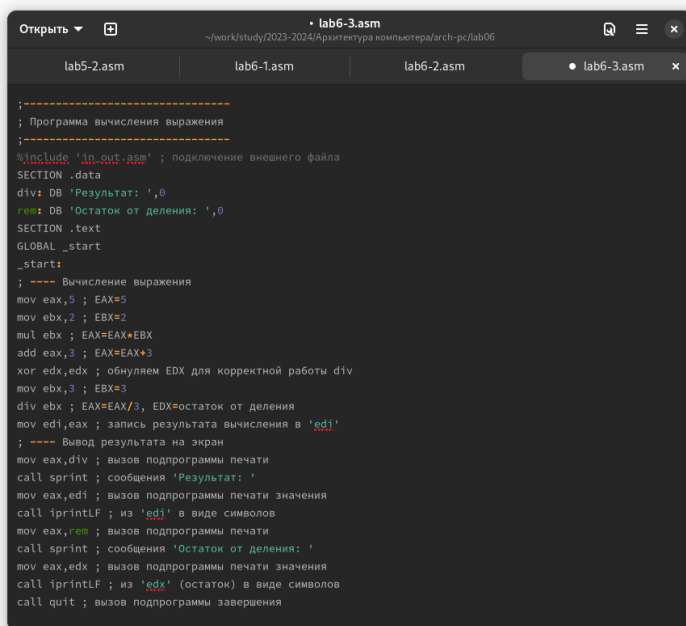
```
[ulyanazaitseva@fedora lab06]$ nasm -f elf lab6-2.asm
[ulyanazaitseva@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[ulyanazaitseva@fedora lab06]$ ./lab6-2
10[ulyanazaitseva@fedora lab06]$
```

### *Проверка*

Вывод функций `iprintLF` и `iprint` отличается тем, что при использовании `iprintLF` выполняется перенос на следующую строку после вывода, а при использовании `iprint` этого не происходит.

## 2. Выполнение арифметических операций в NASM

Создаю файл `lab6-3.asm` и ввожу в него текст из листинга 6.3.(рис. ??)



```
;-----  
; 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,5 ; EAX=5  
mov ebx,2 ; EBX=2  
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  
call iprintf ; 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 iprintf ; из 'edx' (остаток) в виде символов  
call quit ; вызов подпрограммы завершения
```

### Третий листинг

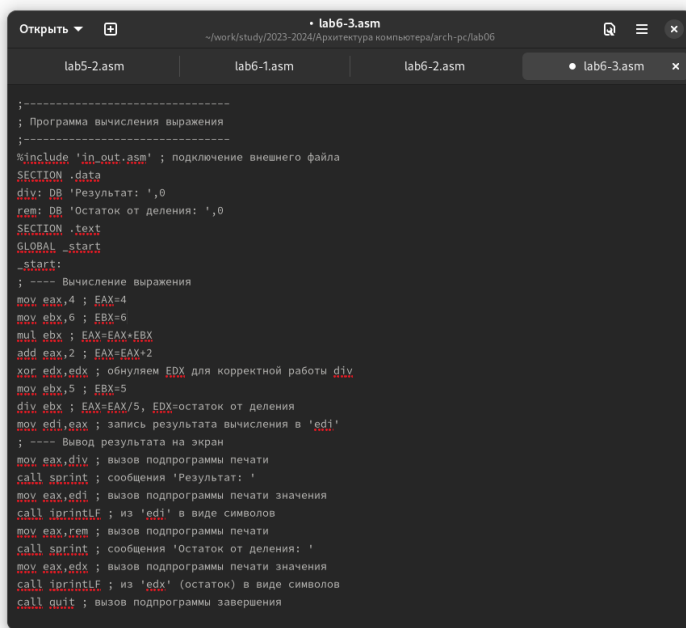
Создаю исполняемый файл и запускаю его.(рис. ??)

```
[ulyanazaitseva@fedora lab06]$ nasm -f elf lab6-3.asm  
[ulyanazaitseva@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o  
[ulyanazaitseva@fedora lab06]$ ./lab6-3  
Результат: 4  
Остаток от деления: 1  
[ulyanazaitseva@fedora lab06]$
```

### Третий листинг

Изменяю текст программы для вычисления выражения  $f(x) = (4 * 6 + 2)/5$ , делая замену чисел в регистрах. (рис. ??)





```
-----  
; 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 sprintf ; сообщения 'Результат: '  
mov eax,edi ; вызов подпрограммы печати значения  
call %printlf ; из 'edi' в виде символов  
mov eax,rem ; вызов подпрограммы печати  
call sprintf ; сообщения 'Остаток от деления: '  
mov eax,edx ; вызов подпрограммы печати значения  
call %printlf ; из 'edx' (остаток) в виде символов  
call quit ; вызов подпрограммы завершения
```

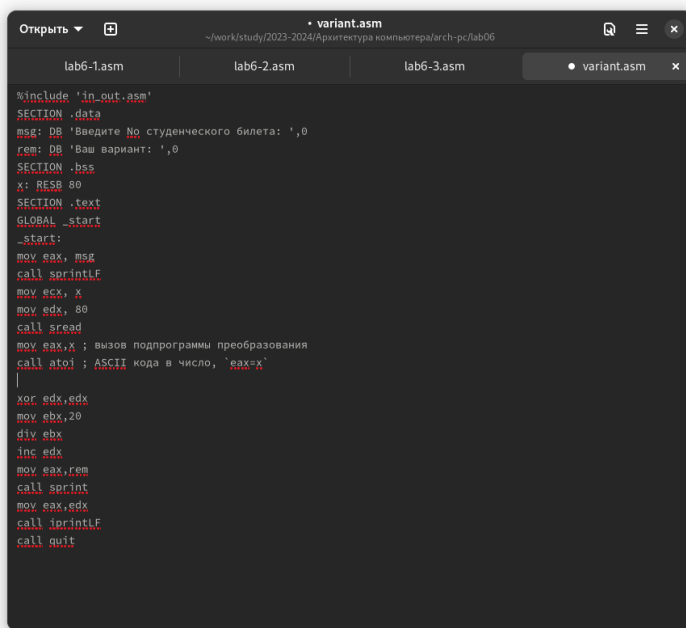
*Изменяю текст программы*

Создаю исполняемый файл и запускаю его.(рис. ??)

```
[ulyanazaitseva@fedora lab06]$ nasm -f elf lab6-3.asm  
[ulyanazaitseva@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o  
[ulyanazaitseva@fedora lab06]$ ./lab6-3  
Результат: 5  
Остаток от деления: 1  
[ulyanazaitseva@fedora lab06]$
```

*Проверяю работу программы*

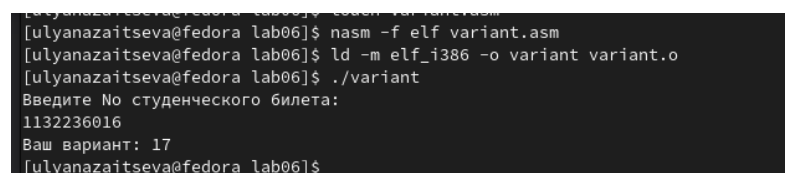
Создаю файл variant.asm. Ввожу текст программы из листинга 6.4, создаю исполняемый файл и запускаю его.(рис. ??)



```
%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 sprintf
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 iprintf
call quit
```

#### Четвертый листинг

Проверяю результат работы программы, вычислив номер варианта.(рис. ??)



```
[ulyanazaitseva@fedora lab06]$ nasm -f elf variant.asm
[ulyanazaitseva@fedora lab06]$ ld -m elf_i386 -o variant variant.o
[ulyanazaitseva@fedora lab06]$ ./variant
Введите No студенческого билета:
1132236016
Ваш вариант: 17
[ulyanazaitseva@fedora lab06]$
```

Вычисление моего варианта по номеру студенческого

3. Ответы на вопросы по листингу 6.4
4. За вывод сообщения “Ваш вариант” отвечают строки кода:

`mov eax,rem`

`call sprint`

2. `mov ecx, x` - закладывает адрес вводимой строки `x` в регистр.

`mov edx, 80` - запись в регистр `edx` длины вводимой строки.

`call sread` - вызов подпрограммы из внешнего файла, обеспечивает ввод сообщения с клавиатуры.

3. ‘`call atoi`’ - вызов подпрограммы из внешнего файла, преобразующей `ascii`-код символа в целое число и записывает результат в регистр `eax`.
4. За вычисления варианта отвечают строки:

```
xor edx,edx
```

```
mov ebx,20
```

```
div ebx
```

```
inc edx
```

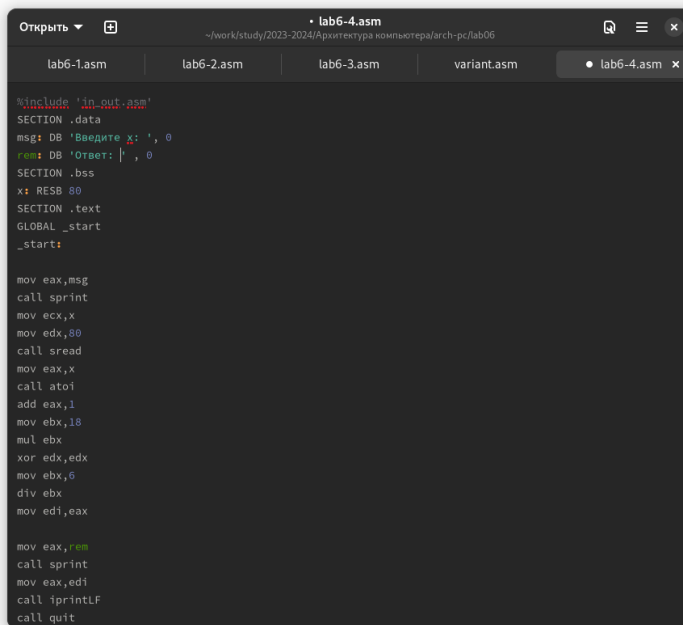
5. При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`.
6. Инструкция `inc edx` увеличивает значение регистра `edx` на 1.
7. За вывод на экран результатов вычислений отвечают строки:

```
mov eax,edx
```

```
call iprintLF
```

4. Задание для самостоятельной работы

Мой варианта- 17, поэтому мне нужно написать программу для вычисления выражения  $18(x + 1)/6$  и проверить ее работу для значений  $x_1 = 3$  и  $x_2 = 1$ . (рис. ??)



```
lab6-4.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06

lab6-1.asm lab6-2.asm lab6-3.asm variant.asm lab6-4.asm x

#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ', 0
fmt: DB 'Ответ: |', 0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:

mov eax,msg
call sprint
mov ecx,x
mov edx,80
call sread
mov eax,x
call atoi
add eax,1
mov ebx,18
mul ebx
xor edx,edx
mov ebx,6
div ebx
mov edi,eax

mov eax,fmt
call sprint
mov eax,edi
call iprintLF
call quit
```

*Программа для вычисления выражения*

Код:

```
%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 sprint
```

```
mov ecx,x
```

```
mov edx,80
```

```
call sread
```

```
mov eax,x
```

```
call atoi
```

```
add eax,1
```

```
mov ebx,18
```

```
mul ebx
```

```
xor edx,edx
```

```
mov ebx,6
```

```
div ebx
```

```
mov edi,eax
```

```
mov eax,rem
```

```
call sprint
```

```
mov eax,edi
```

```
call iprintLF
```

```
call quit
```

Создаю исполняемый файл и проверяю его работу.(рис. ??)

```
0100110
[ulyanazaitseva@fedora lab06]$ nasm -f elf lab6-4.asm
[ulyanazaitseva@fedora lab06]$ ld -m elf_i386 -o lab6-4 lab6-4.o
[ulyanazaitseva@fedora lab06]$ ./lab6-4
Введите x: 3
Ответ: 12
[ulyanazaitseva@fedora lab06]$ nasm -f elf lab6-4.asm
[ulyanazaitseva@fedora lab06]$ ld -m elf_i386 -o lab6-4 lab6-4.o
[ulyanazaitseva@fedora lab06]$ ./lab6-4
Введите x: 1
Ответ: 6
[ulyanazaitseva@fedora lab06]$
```

*Проверка работы со значениями x1, x2*

## 5 Выводы

С помощью данной лабораторной работы я освоила арифметические инструкции языка ассемблер NASM