

# **Отчет по лабораторной работе №6**

**Дисциплина: архитектура компьютера**

Гашимова Эсма Эльшан кызы

# 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>
4.1	Символьные и численные данные в NASM . . . . .	9
4.2	Выполнение арифметических операций в NASM . . . . .	12
4.3	Ответы на контрольные вопросы . . . . .	15
4.4	Задание для самостоятельной работы . . . . .	16
<b>5</b>	<b>Выводы</b>	<b>19</b>
<b>6</b>	<b>Список литературы</b>	<b>20</b>

## Список иллюстраций

4.1	Создание нового каталога . . . . .	9
4.2	Сохранение новой программы . . . . .	10
4.3	Запуск изначальной программы . . . . .	10
4.4	Измененная программа . . . . .	11
4.5	Запуск измененной программы . . . . .	11
4.6	Вторая программа . . . . .	11
4.7	Вывод второй программы . . . . .	12
4.8	Вывод измененной второй программы . . . . .	12
4.9	Замена функции вывода во второй программе . . . . .	12
4.10	Третья программа . . . . .	13
4.11	Запуск третьей программы . . . . .	13
4.12	Изменение третьей программы . . . . .	14
4.13	Запуск измененной третьей программы . . . . .	14
4.14	Программа для подсчета варианта . . . . .	15
4.15	Запуск программы для подсчета варианта . . . . .	15
4.16	Запуск и проверка программы . . . . .	16

## **Список таблиц**

# 1 Цель работы

Цель данной лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

## **2 Задание**

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий 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. - 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. 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. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними

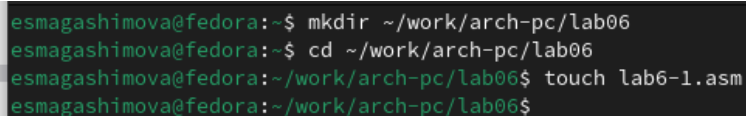
арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно.



## 4 Выполнение лабораторной работы

### 4.1 Символьные и численные данные в NASM

Создаю каталог для программ лабораторной работы №6 и перехожу в него, создаю там файл (рис. 4.1).



```
esmagashimova@fedora:~$ mkdir ~/work/arch-pc/lab06
esmagashimova@fedora:~$ cd ~/work/arch-pc/lab06
esmagashimova@fedora:~/work/arch-pc/lab06$ touch lab6-1.asm
esmagashimova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.1: Создание нового каталога

В созданном файле ввожу программу из листинга (рис. 4.2).

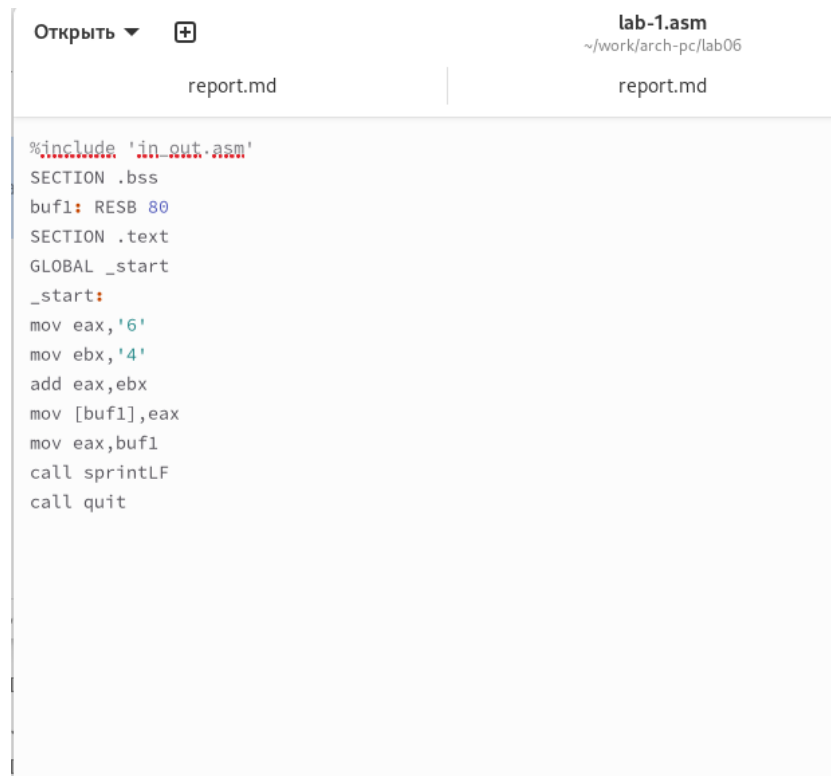


Рис. 4.2: Сохранение новой программы

Создаю исполняемый файл и запускаю его, вывод программы отличается от предполагаемого изначально, так как коды символов в сумме дают символ j по таблице ASCII. (рис. 4.3)

```

esmagashimova@fedora: ~/work/arch-pc/lab06$ mousepad lab6-1.asm
esmagashimova@fedora: ~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
esmagashimova@fedora: ~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
esmagashimova@fedora: ~/work/arch-pc/lab06$ ./lab6-1
j
esmagashimova@fedora: ~/work/arch-pc/lab06$

```

Рис. 4.3: Запуск изначальной программы

Изменяю текст изначальной программы, убрав кавычки (рис. 4.4).

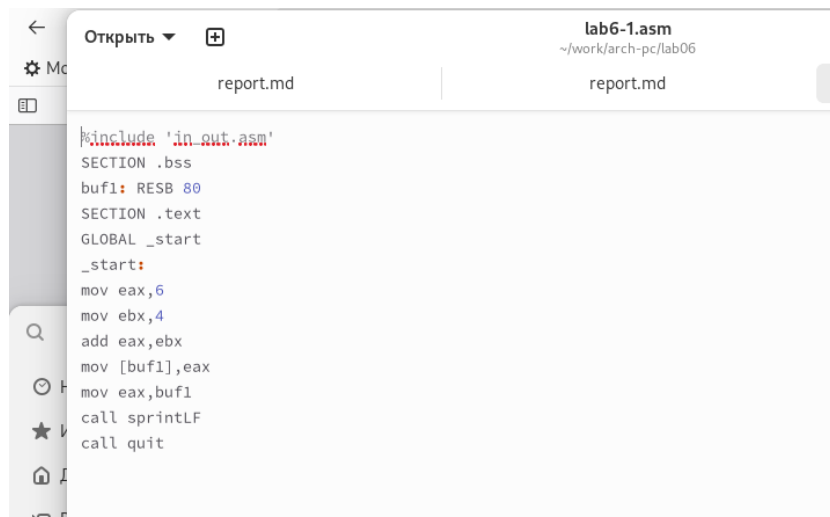


Рис. 4.4: Измененная программа

На этот раз программа выдала пустую строку, это потому что символ 10 означает переход на новую строку (рис. 4.5).

```

esmagashimova@fedora:~/work/arch-pc/lab06$ mousepad lab6-1.asm
esmagashimova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
esmagashimova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
esmagashimova@fedora:~/work/arch-pc/lab06$ ./lab6-1

esmagashimova@fedora:~/work/arch-pc/lab06$

```

Рис. 4.5: Запуск измененной программы

Создаю новый файл для будущей программы и записываю в нее код из листинга (рис. 4.6).

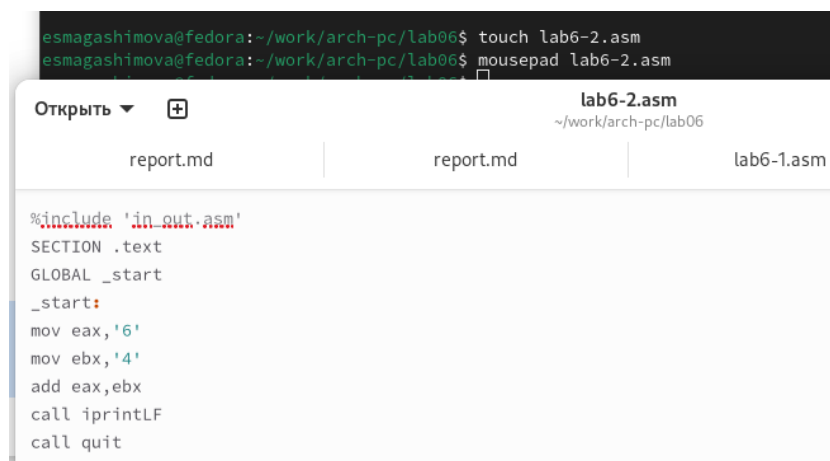


Рис. 4.6: Вторая программа

Создаю исполняемый файл и запускаю его, теперь отображается результат 106, программа, как и в первый раз, сложила коды символов, но вывела само число, а не его символ, благодаря замене функции вывода на `iprintLF` (рис. 4.7).

```
esmagashimova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
esmagashimova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
esmagashimova@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
esmagashimova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.7: Вывод второй программы

Убрав кавычки в программе, я снова ее запускаю и получаю предполагаемый изначально результат. (рис. 4.8).

```
esmagashimova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
esmagashimova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
esmagashimova@fedora:~/work/arch-pc/lab06$ ./lab6-2
10
esmagashimova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.8: Вывод измененной второй программы

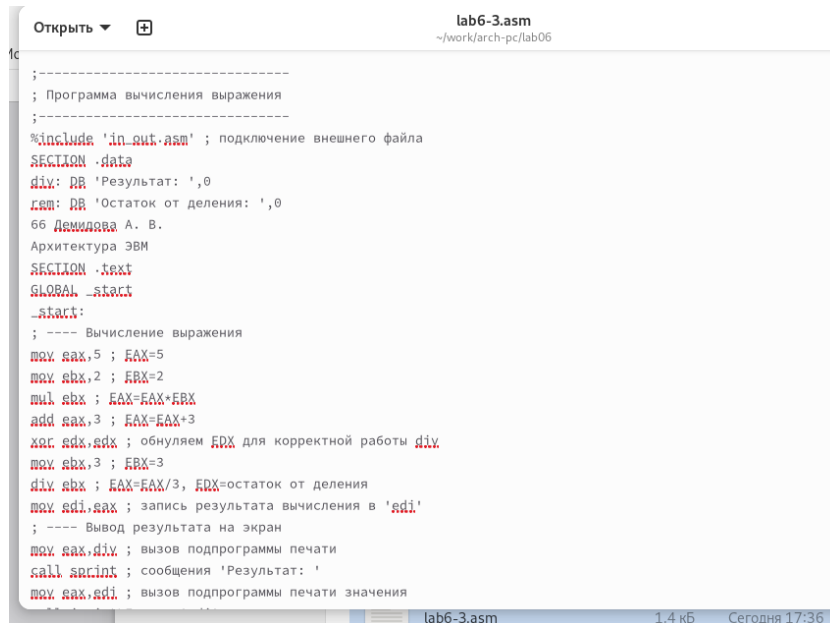
Заменив функцию вывода на `iprint`, я получаю тот же результат, но без переноса строки (рис. 4.9).

```
esmagashimova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
esmagashimova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
esmagashimova@fedora:~/work/arch-pc/lab06$ ./lab6-2
10esmagashimova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.9: Замена функции вывода во второй программе

## 4.2 Выполнение арифметических операций в NASM

Создаю новый файл и копирую в него содержимое листинга (рис. 4.10).

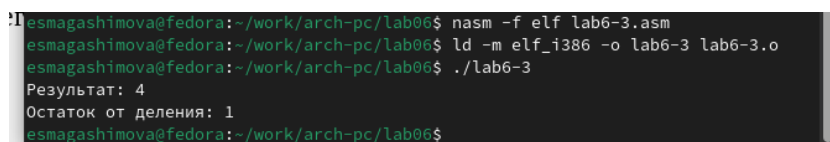


```
Открыть + lab6-3.asm
~/work/arch-pc/lab06

; -----
; Программа вычисления выражения
; -----
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
66 Демидова А. В.
Архитектура ЭВМ
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 sprintf ; сообщения 'Результат: '
mov eax,rem ; вызов подпрограммы печати значения
```

Рис. 4.10: Третья программа

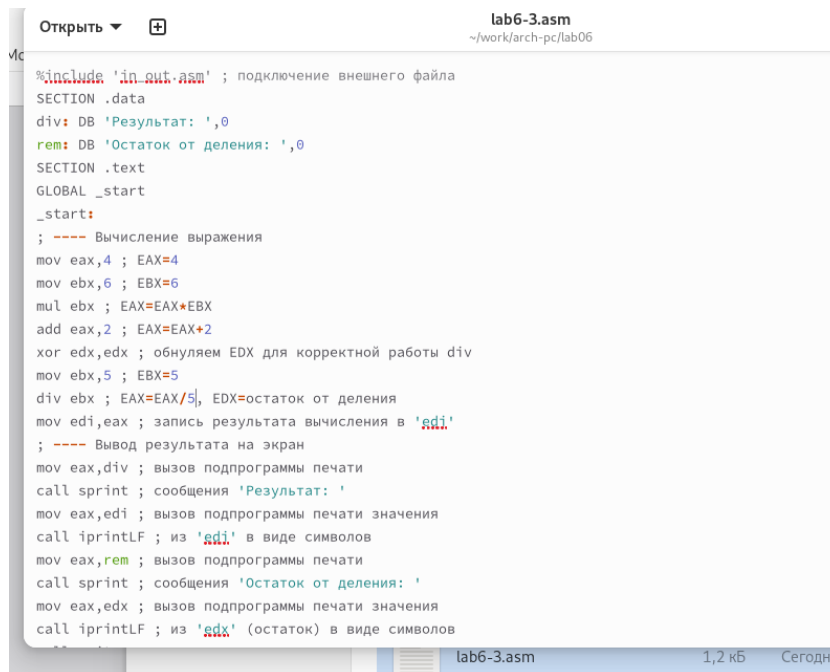
Программа выполняет арифметические вычисления, на вывод идет результирующее выражения и его остаток от деления (рис. 4.11).



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

Рис. 4.11: Запуск третьей программы

Заменяя переменные в программе для выражения  $f(x) = (4*6+2)/5$  (рис. 4.12).




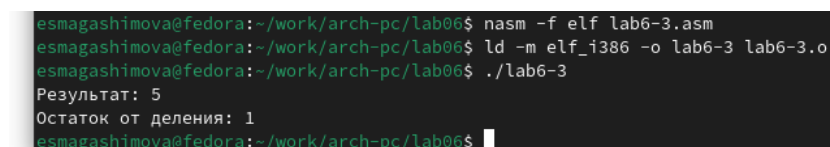
```
Открыть ▾  lab6-3.asm  
~/work/arch-pc/lab06  
%include 'in_out.asm' ; подключение внешнего файла  
SECTION .data  
div: DB 'Результат: ',0  
rem: DB 'Остаток от деления: ',0  
SECTION .text  
GLOBAL _start  
_start:  
; ---- Вычисление выражения  
mov eax,4 ; EAX=4  
mov ebx,6 ; EBX=6  
mul ebx ; 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
```

Рис. 4.12: Изменение третьей программы

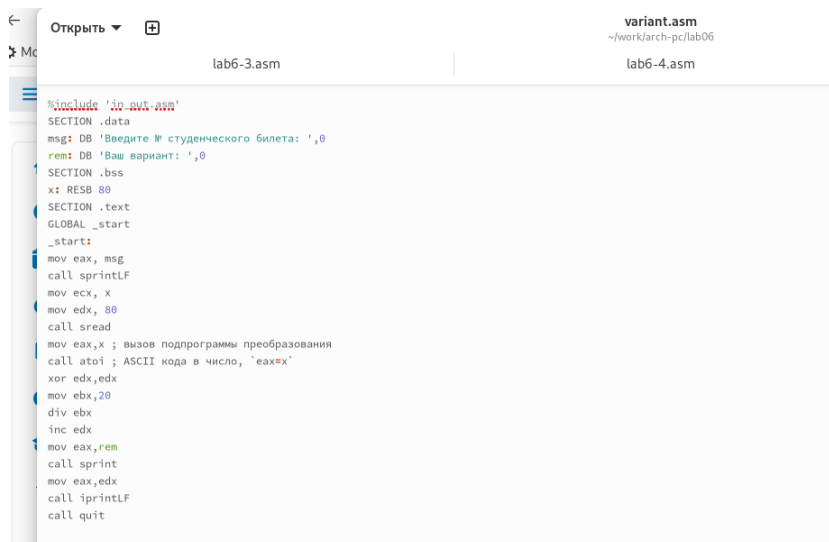
Запуск программы дает корректный результат (рис. 4.13).



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

Рис. 4.13: Запуск измененной третьей программы

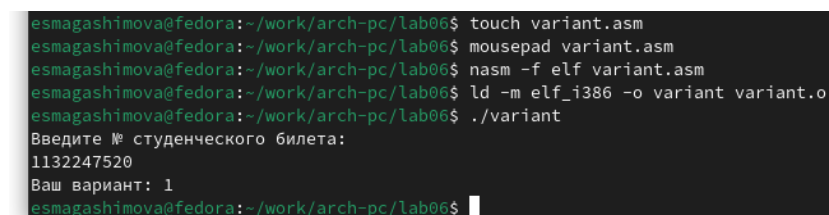
Создаю новый файл и помещаю текст из листинга (рис. 4.14).



```
%include 'io_out.asm'
SECTION .data
msg: DB 'Введите № 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
```

Рис. 4.14: Программа для подсчета варианта

Запустив программу и указав свой номер студенческого билета, я получил свой вариант для дальнейшей работы. (рис. 4.15).



```
esmagashimova@fedora: ~/work/arch-pc/lab06$ touch variant.asm
esmagashimova@fedora: ~/work/arch-pc/lab06$ mousepad variant.asm
esmagashimova@fedora: ~/work/arch-pc/lab06$ nasm -f elf variant.asm
esmagashimova@fedora: ~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
esmagashimova@fedora: ~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132247520
Ваш вариант: 1
esmagashimova@fedora: ~/work/arch-pc/lab06$
```

Рис. 4.15: Запуск программы для подсчета варианта

## 4.3 Ответы на контрольные вопросы

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

```
mov eax, rem
call sprint
```

2. Инструкция `mov ecx, x` используется, чтобы положить адрес вводимой строки `x` в регистр `ecx` `mov edx, 80` - запись в регистр `edx` длины вводимой строки `call sread` - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры.

3. `call atoi` используется для вызова подпрограммы из внешнего файла, которая преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`.

4. За вычисления варианта отвечают строки:

```
xor edx,edx ; обнуление edx для корректной работы div
mov ebx,20 ; ebx = 20
div ebx ; eax = eax/20, edx - остаток от деления
inc edx ; edx = edx + 1
```

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
```

## 4.4 Задание для самостоятельной работы

В соответствии с выбранным вариантом, я реализую программу для подсчета функции  $f(x) = (10 + 2x)/3$ , проверка на нескольких переменных показывает корректное выполнение программы (рис. 4.16).

```
esmagashimova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
esmagashimova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
ld: невозможно найти lab6-4.0: Нет такого файла или каталога
esmagashimova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
esmagashimova@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 4esmagashimova@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 10
Результат: 10esmagashimova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.16: Запуск и проверка программы

Прилагаю код своей программы:



```
%include 'in_out.asm' ; подключение внешнего файла с процедурами
```

## SECTION .data

```
msg db 'Введите значение переменной x: ', 0 ; Сообщение для запроса ввода
rem db 'Результат: ', 0 ; Сообщение для вывода результата
exp_msg db 'y = (2 * x + 10) / 3', 0 ; Формула, которую можно вывести
```

## SECTION .bss

```
x resb 80 ; Переменная для ввода значения x
```

## SECTION .text

```
GLOBAL _start ; Точка входа в программу
```

**\_start:**

```
; ---- Выводим сообщение "Введите значение переменной x: "
```

```
mov eax, msg
```

```
call sprint ; Вызов подпрограммы печати запроса
```

```
; ---- Читаем ввод пользователя (значение x)
```

```
mov ecx, x ; Адрес переменной x
```

```
mov edx, 80 ; Максимальная длина ввода
```

```
call sread ; Ввод значения с клавиатуры
```

```
; ---- Преобразуем строку в число
```

```
mov eax, x ; Адрес строки с введённым значением
```

```
call atoi ; Преобразуем строку в число, eax = x
```

```
; ---- Вычисляем выражение (2 * x + 10) / 3
```

```
mov ebx, 2 ; Множитель 2
```

```

imul eax, ebx ;  $eax = 2 * x$ 

add eax, 10    ;  $eax = (2 * x) + 10$ 

mov ebx, 3     ; Делитель 3
xor edx, edx ; Очищаем edx перед делением
div ebx        ;  $eax = (2 * x + 10) / 3$ , результат в eax

; ---- Результат вычисления в eax, сохраняем в переменную result
mov edi, eax ; Сохраняем результат в edi

; ---- Выводим сообщение "Результат: "
mov eax, rem
call sprint   ; Выводим сообщение "Результат: "

; ---- Печатаем результат
mov eax, edi ; Загружаем результат из edi
call iprint  ; Печатаем результат как целое число

; ---- Завершаем программу
call quit    ; Завершаем программу

```

## **5 Выводы**

При выполнении данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

## **6 Список литературы**

1. Курс на ТУИС
2. Лабораторная работа №6
3. Программирование на языке ассемблера NASM Столяров А. В.