

# ***Отчёт по лабораторной работе***

***Лабораторная №6***

Полина Витальевна Барабаш

# ***Содержание***

<b><i>1</i></b>	<b><i>Цель работы</i></b>	<b><i>4</i></b>
<b><i>2</i></b>	<b><i>Выполнение работы</i></b>	<b><i>5</i></b>
<b><i>3</i></b>	<b><i>Выполнение самостоятельной работы</i></b>	<b><i>13</i></b>
<b><i>4</i></b>	<b><i>Выводы</i></b>	<b><i>16</i></b>

## ***Список иллюстраций***

2.1	Создание каталога и файла внутри него . . . . .	5
2.2	Создание исполняемого файла и запуск программы . . . . .	6
2.3	Создание исполняемого файла и запуск изменённой программы .	6
2.4	Создание файла с текстом программы, создание исполняемого файла и запуск программы . . . . .	7
2.5	Создание исполняемого файла и запуск изменённой программы .	8
2.6	Работа программы с функцией <code>iprint</code> вместо <code>iprintLF</code> . . . . .	8
2.7	Создание файла <code>lab6-3.asm</code> , создание исполняемого файла после записи текста программы и запуск программы . . . . .	9
2.8	Работа программы по вычислению функции с другими значениями	9
2.9	Создание, запуск и работа программы, вычисляющей вариант по номеру студенческого билета . . . . .	10
3.1	Создание, запуск и работа программы по вычислению данной функции . . . . .	15

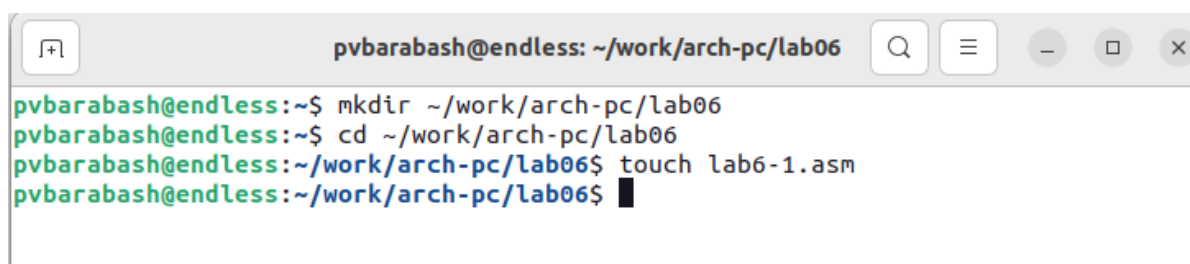
# ***1 Цель работы***

Освоение арифметических инструкций языка ассемблера NASM.

## 2 Выполнение работы

**Задание №1.** Создайте каталог для программ лабораторной работы № 6, перейдите в него и создайте файл lab6-1.asm.

Я создала каталог lab06 с помощью `mkdir ~/work/arch-pc/lab06`. Затем перешла в него с помощью `cd ~/work/arch-pc/lab06`. И создала файл с помощью `touch lab6-1.asm` (рис. 2.1).

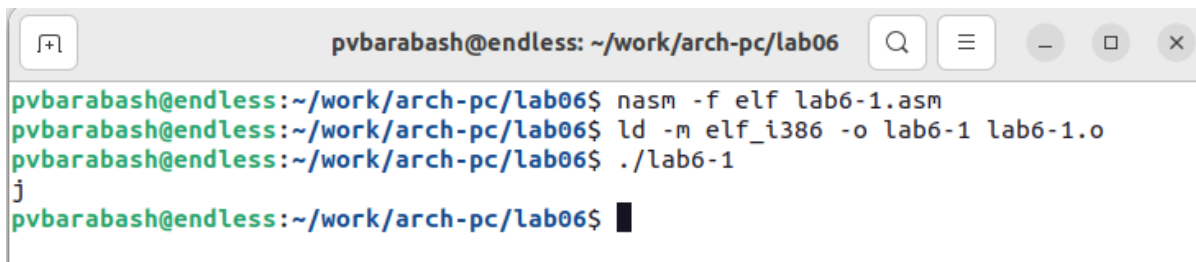
A screenshot of a terminal window. The title bar shows the user 'pvbarabash@endless' and the current directory '~/work/arch-pc/lab06'. The terminal contains the following commands and their outputs:

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

Рис. 2.1: Создание каталога и файла внутри него

**Задание №2.** Введите в файл lab6-1.asm текст программы из листинга 6.1. Создайте исполняемый файл и запустите его.

Я открыла lab6-1.asm в Midnight Commander и ввела в файл текст программы из листинга 6.1. Я создала исполняемый файл и запустила его (рис. 2.2).



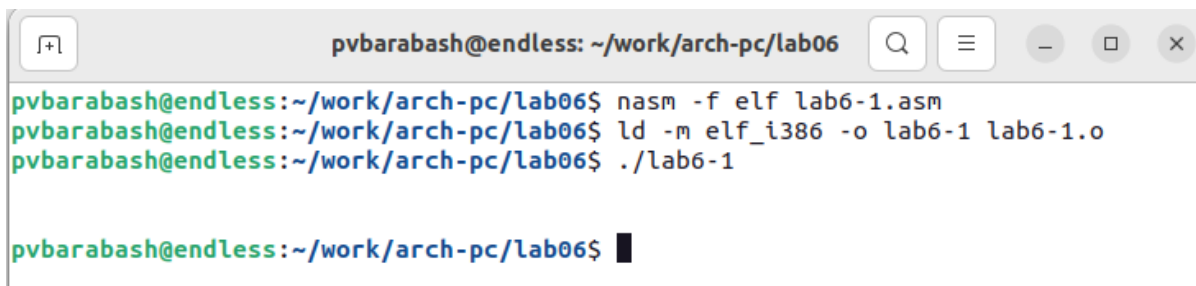
```
pvbarabash@endless: ~/work/arch-pc/lab06
pvbarabash@endless:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
pvbarabash@endless:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
pvbarabash@endless:~/work/arch-pc/lab06$ ./lab6-1
j
pvbarabash@endless:~/work/arch-pc/lab06$
```

Рис. 2.2: Создание исполняемого файла и запуск программы

На (рис. 2.2) видно, что программа вывела символ “j”, как и описано в руководстве по выполнению лабораторной работы №6. Это происходит из-за того, что мы вводим символы ‘6’ и ‘4’, которые в десятичном представлении имеют коды 54 и 52 соответственно. Поэтому при сложении получается код 106, который соответствует по таблице ASCII символу ‘j’.

**Задание №3.** Измените текст программы и вместо символов запишите в регистры числа. Создайте исполняемый файл и запустите его. Пользуясь таблицей ASCII определите какому символу соответствует код 10 (который получается при сложении). Отображается ли этот символ при выводе на экран?

Я изменила текст программы и вместо символов записала в регистры числа, как дано в тексте. Я создала исполняемый файл и запустила его, получив следующую выдачу, см. (рис. 2.3).



```
pvbarabash@endless: ~/work/arch-pc/lab06
pvbarabash@endless:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
pvbarabash@endless:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
pvbarabash@endless:~/work/arch-pc/lab06$ ./lab6-1

pvbarabash@endless:~/work/arch-pc/lab06$
```

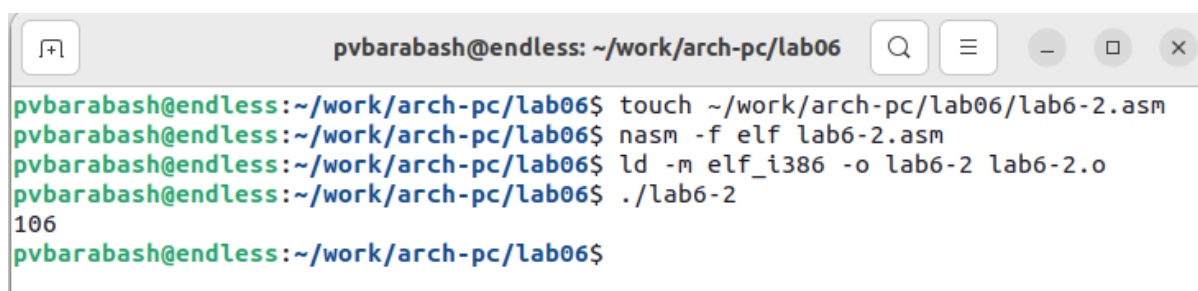
Рис. 2.3: Создание исполняемого файла и запуск изменённой программы

По таблице ASCII код 10 имеет символ “LF, \n”, то есть перенос строки. На (рис. 2.3) видно, что действительно результатом программы является перенос строки,

то есть символ отображается.

**Задание №4.** Преобразовать текст программы из Листинга 6.1 с использованием подпрограмм для преобразования ASCII символов в числа и обратно из файла `in_out.asm`. Создайте файл `lab6-2.asm` в каталоге `~/work/arch-pc/lab06` и введите в него текст программы из листинга 6.2. Создайте исполняемый файл и запустите его.

Я создала файл `lab6-2.asm` с помощью команды `touch`. Открыла `lab6-2.asm` в Midnight Commander и ввела в файл текст программы из листинга 6.2. Я создала исполняемый файл и запустила его (рис. 2.4).



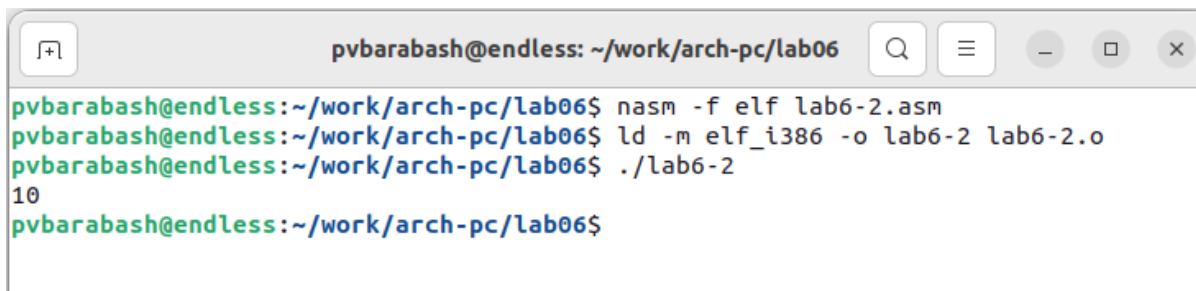
```
pvbarabash@endless: ~/work/arch-pc/lab06
pvbarabash@endless:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
pvbarabash@endless:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
pvbarabash@endless:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
pvbarabash@endless:~/work/arch-pc/lab06$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

Рис. 2.4: Создание файла с текстом программы, создание исполняемого файла и запуск программы

Программа вывела 106, сложив коды символов '6' и '4', как в программе из задания 1. Однако, благодаря функции `iprintLF`, выводится не символ 'j', чьим кодом является 106, а число 106.

**Задание №5.** Аналогично предыдущему примеру измените символы на числа. Создайте исполняемый файл и запустите его. Какой результат будет получен при исполнении программы? Замените функцию `iprintLF` на `iprint`. Создайте исполняемый файл и запустите его. Чем отличается вывод функций `iprintLF` и `iprint`?

Аналогично предыдущему примеру я изменила символы на числа. Создала исполняемый файл и запустила его. Программа выдала в ответ 10, тот ответ, который мы и хотели получить (рис. 2.5).

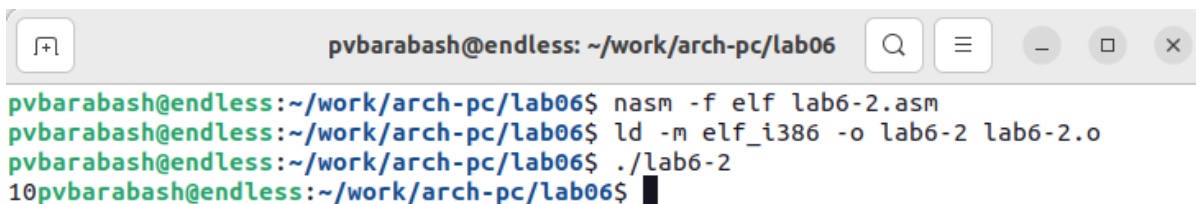


```
pvbarabash@endless: ~/work/arch-pc/lab06
pvbarabash@endless:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
pvbarabash@endless:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
pvbarabash@endless:~/work/arch-pc/lab06$ ./lab6-2
10
pvbarabash@endless:~/work/arch-pc/lab06$
```

Рис. 2.5: Создание исполняемого файла и запуск изменённой программы

Аналогично предыдущему примеру мы получаем 10 при сложении 6 и 4, но теперь благодаря функции `iprintLF`, выводится не соответствующий коду символ, а само число 10.

Я заменила функцию `iprintLF` на `iprint`. Я создала исполняемый файл и запустила его (рис. 2.6).



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

Рис. 2.6: Работа программы с функцией `iprint` вместо `iprintLF`

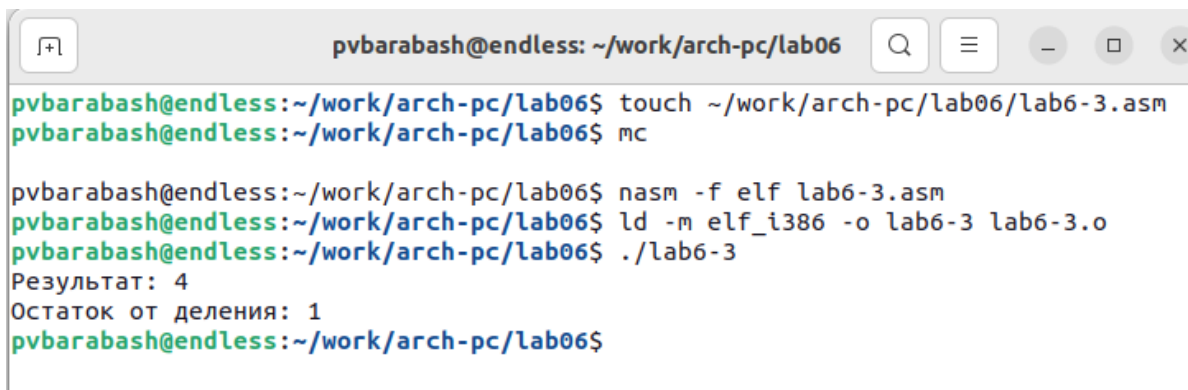
Как можно видеть на (рис. 2.6) результат получен такой же – 10, но отличие в отсутствии перевода строки после результата.

**Задание №6.** В качестве примера выполнения арифметических операций в NASM приводится программа вычисления арифметического выражения  $f(x) = (5 * 2 + 3)/3$ . Создайте файл `lab6-3.asm` в каталоге `~/work/arch-pc/lab06`. Внимательно изучите текст программы из листинга 6.3 и введите в `lab6-3.asm`. Создайте исполняемый файл и запустите его.

Я создала файл `lab6-3.asm` в каталоге `~/work/arch-pc/lab06` с помощью команды



touch. Внимательно изучила текст программы из листинга 6.3 и ввела его в lab6-3.asm. Я создала исполняемый файл и запустила его (рис. 2.7).



```
pvbarabash@endless: ~/work/arch-pc/lab06
pvbarabash@endless:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
pvbarabash@endless:~/work/arch-pc/lab06$ mc

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

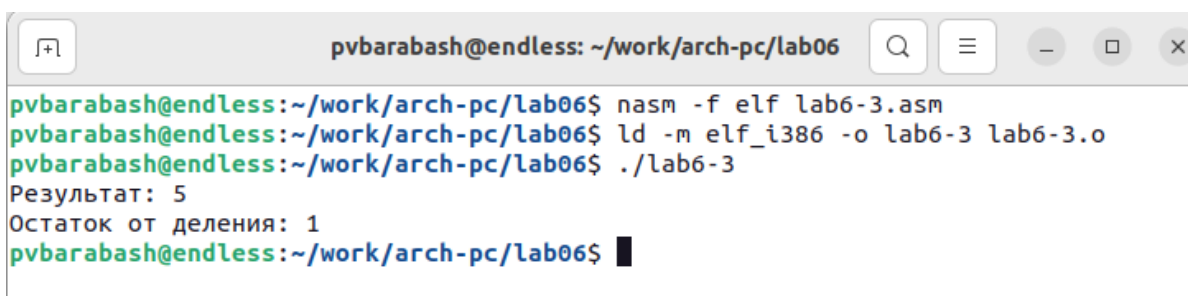
Рис. 2.7: Создание файла lab6-3.asm, создание исполняемого файла после записи текста программы и запуск программы

На (рис. 2.7) видно, что программа выдаёт необходимый результат, зафиксированный в тексте рекомендаций по выполнению лабораторной работы №6.

**Задание №7.** Измените текст программы для вычисления выражения  $f(x) = (4 * 6 + 2)/5$ . Создайте исполняемый файл и проверьте его работу.

Я изменила текст программы для вычисления новой функции. Так как функция идентична функции из предыдущего задания, необходимо только заменить числа в предыдущей программе на данные. Файл с изменённой программой будет прикреплен в ТУИС.

После изменения кода, я создала исполняемый файл и проверила его работу (рис. 2.8).



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

Рис. 2.8: Работа программы по вычислению функции с другими значениями

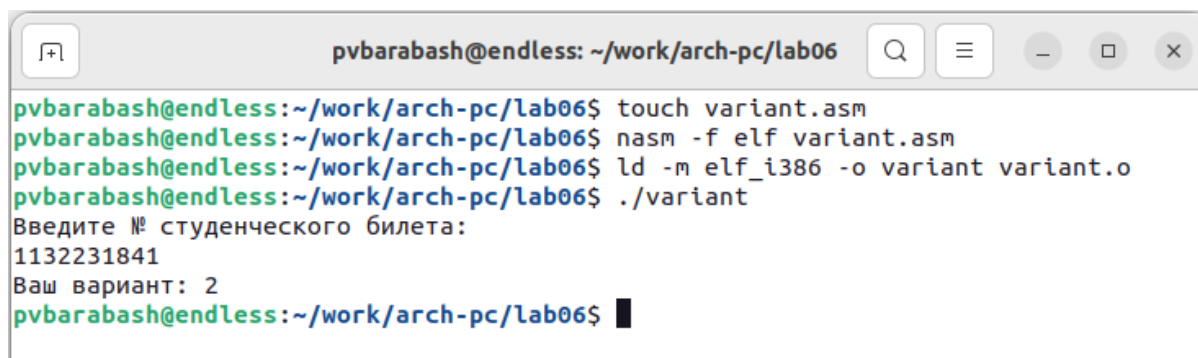
Программа выдаёт результат 5 и остаток 1.

Действительно  $4 * 6 = 24$ ;  $24 + 2 = 26$ ;  $26 / 5 = 5$  (и остаток от деления 1). Программа работает верно.

**Задание №9.** Создайте файл `variant.asm` в каталоге `~/work/arch-pc/lab06`. Внимательно изучите текст программы из листинга 6.4 и введите в файл `variant.asm`. Создайте исполняемый файл и запустите его. Проверьте результат работы программы, вычислив номер варианта аналитически.

Включите в отчет по выполнению лабораторной работы ответы на вопросы.1

Я создала файл `variant.asm` в каталоге `~/work/arch-pc/lab06` с помощью команды `touch`. Внимательно изучила текст программы из листинга 6.4 и ввела в файл `variant.asm`. Затем я создала исполняемый файл и запустила его (рис. 2.9).



```
pvbarabash@endless: ~/work/arch-pc/lab06
pvbarabash@endless:~/work/arch-pc/lab06$ touch variant.asm
pvbarabash@endless:~/work/arch-pc/lab06$ nasm -f elf variant.asm
pvbarabash@endless:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
pvbarabash@endless:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132231841
Ваш вариант: 2
pvbarabash@endless:~/work/arch-pc/lab06$
```

Рис. 2.9: Создание, запуск и работа программы, вычисляющей вариант по номеру студенческого билета

Я ввела номер своего студенческого билета — 1132231841. Моим вариантом программа вывела 2. Проверим правильность выполнения ручными расчётами:  $1132231841 / 20 = 56611592$  и 1 в остатке. Затем к 1 прибавляем 1. Получается 2, как и выдала программа.

Ответы на вопросы:

**1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?**

Следующие строки отвечают за вывод на экран сообщения ‘Ваш вариант’:

```
mov eax,rem
```

```
call sprint
```

В `rem` записано необходимое сообщение, чтобы функция вывода правильно работала, необходимо переместить значение `rem` в `eax`. Функция `sprint` — функция вывода **сообщения** на экран.

## **2. Для чего используются следующие инструкции?**

```
mov ecx, x
```

```
mov edx, 80
```

```
call sread
```

Приведённые инструкции используются для ввода сообщения с клавиатуры. Переменная `x` — буфер размером 80 байт. Сначала мы записываем его в регистр, чтобы передать адрес переменной, в которую будет записано введенное сообщение. Второй строчкой введена длина строки. Третья — вызов подпрограммы, которая считывает сообщение с клавиатуры.

## **3. Для чего используется инструкция “call atoi”?**

Эта функция преобразует `ascii`-код символа в целое число.

## **4. Какие строки листинга 6.4 отвечают за вычисления варианта?**

Следующие строки отвечают за вычисление варианта:

```
xor edx,edx
```

```
mov ebx,20
```

```
div ebx
```

```
inc edx
```

Первой строчкой обнуляем `EDX` для корректной работы `div`. Второй присваиваем `EBX` значение 20 (наш делитель). Третьей делим `EAX` (в котором находится номер студенческого билета после использования функции `atoi`) на `EBX` (то есть на 20). Остаток записывается в регистр `EDX`. Последняя строчка прибавляет к `EDX` единицу.

## **5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?**

Остаток от деления записывается в регистр EDX, поэтому единица прибавляется именно к EDX (`inc edx`).

#### **6. Для чего используется инструкция “`inc edx`”?**

Инструкция `inc edx` прибавляет к остатку, записанному в EDX, единицу.

#### **7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?**

Следующие строки отвечают за вывод на экран результата вычислений:

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

Про вывод сообщения “Ваш вариант” написано при ответе на вопрос №1. После этого мы передаём в регистр EAX значение переменной EDX (остаток от деления + единица). Оно выводится функцией `iprintLF`, которая после вывода **числа в формате ASCII** переводит на новую строку.

### ***3 Выполнение самостоятельной работы***

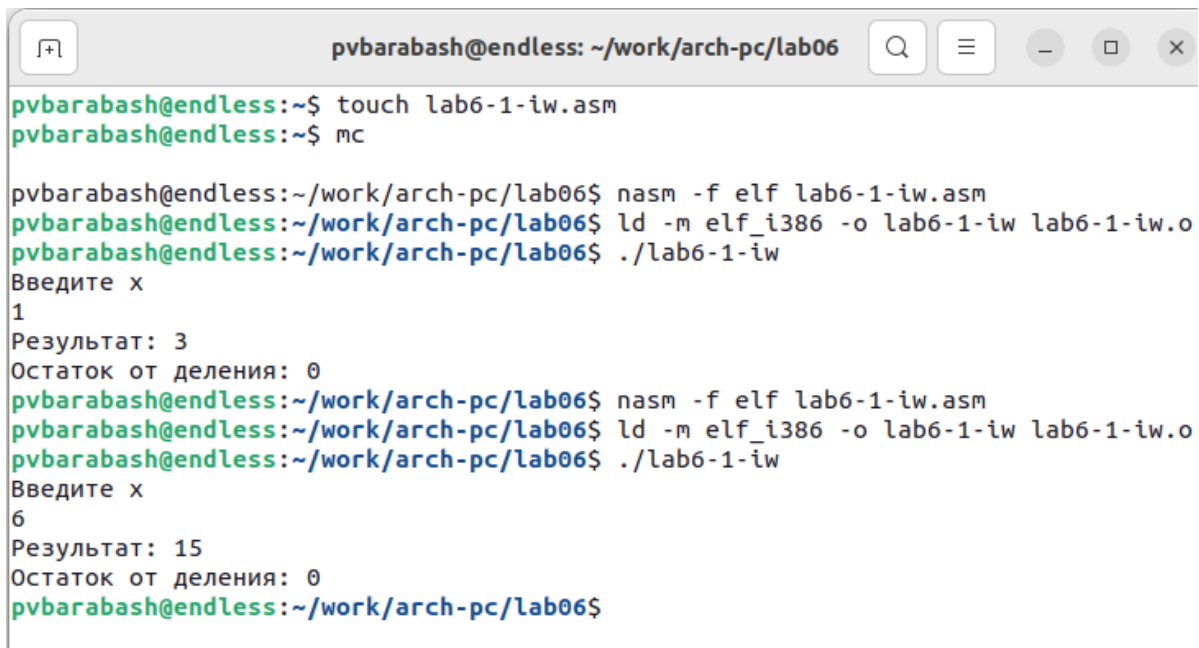
**Задание №1.** Написать программу вычисления выражения  $y = f(x)$ . Программа должна выводить выражение для вычисления, выводить запрос на ввод значения  $x$ , вычислять заданное выражение в зависимости от введенного  $x$ , выводить результат вычислений. Вид функции  $f(x)$  выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений  $x_1$  и  $x_2$  из 6.3.

Я получила второй вариант после выполнения программы из задания №9. Функция в таблице выглядит так:  $(12x + 3)5$ , посчитать нужно для  $x_1 = 1$ ,  $x_2 = 6$ . К сожалению, в функции похоже допущена опечатка. Из-за схожести на функцию из первого варианта, я решила принять, что пропущен знак деления. Поэтому мной будет написана программа по вычислению  $f(x) = (12x + 3)/5$ . Я создала файл lab6-1-iw.asm и написала в нём следующий текст программы:

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x',0
div: DB 'Результат:',0
rem: DB 'Остаток от деления:',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
```

```
mov eax, msg
call sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov ebx, 12
mul ebx
add eax, 3
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
```

Я создала исполняемый файл и запустила программу. При  $x_1 = 1$  ответ равен 3, при  $x_2 = 6$  ответ равен 15 (рис. 3.1).

A terminal window titled 'pvbarabash@endless: ~/work/arch-pc/lab06'. The user enters 'touch lab6-1-iw.asm' and 'mc'. Then they compile with 'nasm -f elf lab6-1-iw.asm', link with 'ld -m elf\_i386 -o lab6-1-iw lab6-1-iw.o', and run './lab6-1-iw'. The program prompts 'Введите x' and the user enters '1'. It outputs 'Результат: 3' and 'Остаток от деления: 0'. The user repeats the compilation and execution, entering '6' and getting 'Результат: 15' and 'Остаток от деления: 0'.

```
pvbarabash@endless:~/work/arch-pc/lab06$ touch lab6-1-iw.asm
pvbarabash@endless:~$ mc

pvbarabash@endless:~/work/arch-pc/lab06$ nasm -f elf lab6-1-iw.asm
pvbarabash@endless:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1-iw lab6-1-iw.o
pvbarabash@endless:~/work/arch-pc/lab06$ ./lab6-1-iw
Введите x
1
Результат: 3
Остаток от деления: 0
pvbarabash@endless:~/work/arch-pc/lab06$ nasm -f elf lab6-1-iw.asm
pvbarabash@endless:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1-iw lab6-1-iw.o
pvbarabash@endless:~/work/arch-pc/lab06$ ./lab6-1-iw
Введите x
6
Результат: 15
Остаток от деления: 0
pvbarabash@endless:~/work/arch-pc/lab06$
```

Рис. 3.1: Создание, запуск и работа программы по вычислению данной функции

Проверим результаты вручную:

$12 * 1 = 12$ ;  $12 + 3 = 15$ ;  $15 / 5 = 3$  (ответ верный)

$12 * 6 = 72$ ;  $72 + 3 = 75$ ;  $75 / 5 = 15$  (ответ верный)

Программа работает правильно.

## **4 Выводы**

Я освоила арифметические инструкции языка ассемблера NASM. Узнала, как оперировать числами, чтобы получать нужный результат, а не перевод в символы ASCII по коду. Освоила арифметические операции сложения, умножения, деления и прибавления единицы. Узнала про вычитание (в том числе единицы).