Отчёт по лабораторной работе №6

диспилина: архитектура компьютера

Терещенкова Маргарита Владимировна

Содержание

# 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. 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

1. Создаю каталог для программам лабораторной работы № 6 (lab06) с помощью утилиты **mkdir**, перехожу в него и создаю файл lab6-1.asm с помощью утилиты **touch**.

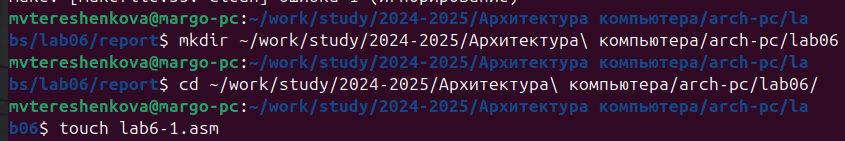


Рис. 1: Создание директории и файла

Копирую в текущий каталог файл in\_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах. И проверяю наличие файла в данной директори с помощью команды **ls**.

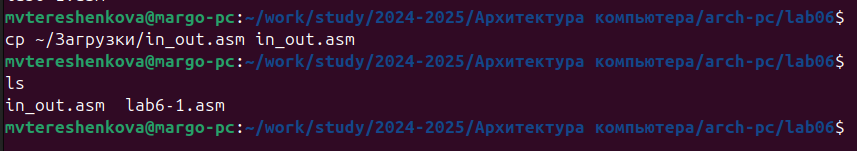


Рис. 2: Копирование файла

1. Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax.

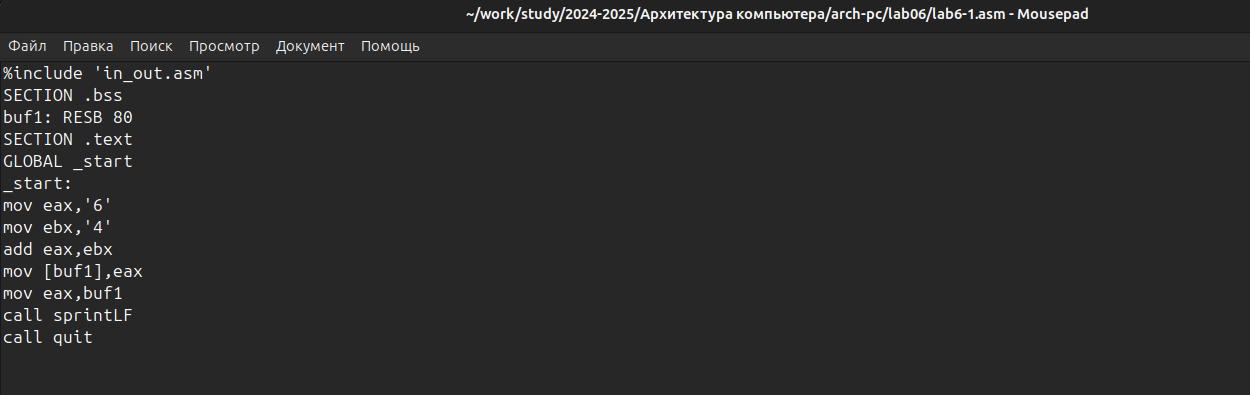


Рис. 3: Редактирование файла

Создаю исполняемый файл программы и запускаю его.

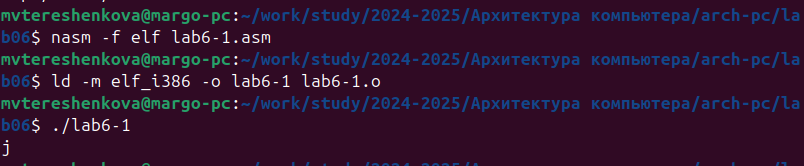


Рис. 4: Запуск исполняемого файла

Вывод программы: символ **j**, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

1. Изменяю текст программы и вместо символов, запишем в регистры числа (“6” и “4” на цифры 6 и 4).

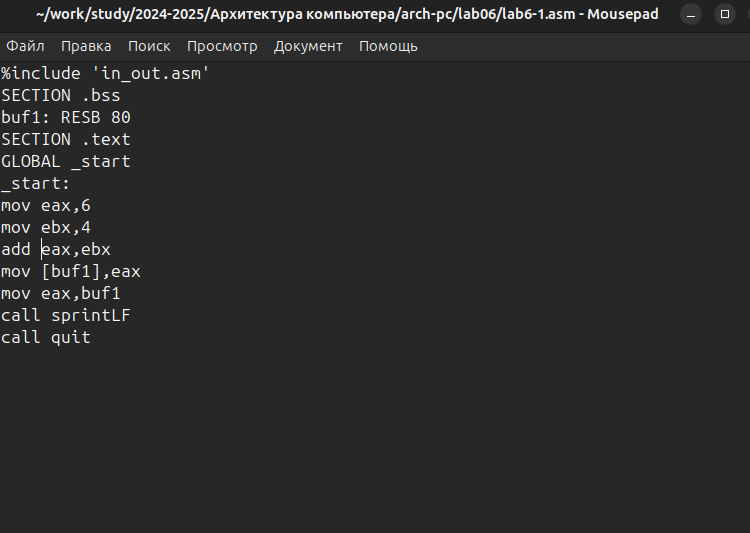


Рис. 5: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его.

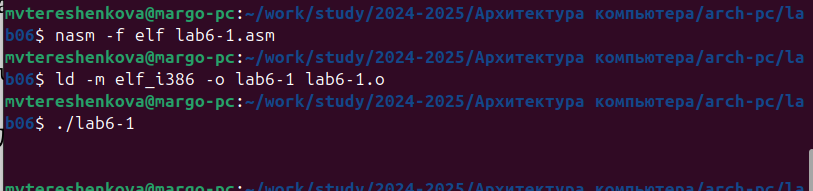


Рис. 6: Запуск исполняемого файла

Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

1. Создаю новый файл lab6-2.asm с помощью утилиты **touch**.

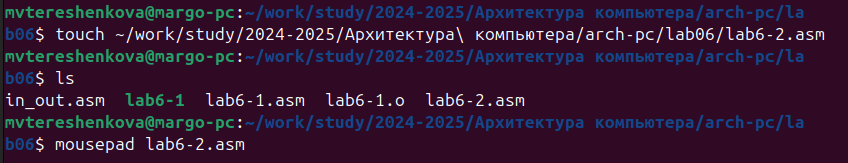


Рис. 7: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax.

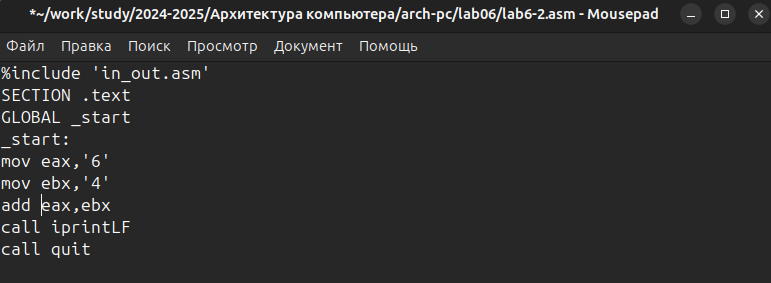


Рис. 8: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2.

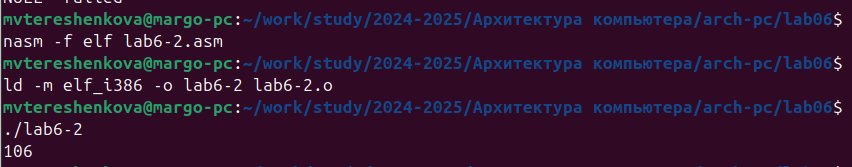


Рис. 9: Запуск исполняемого файла

Теперь выводится число **106**, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.

1. Аналогично предыдущему примеру изменила символы на числа. (“6” и “4” на 6 и 4 ).

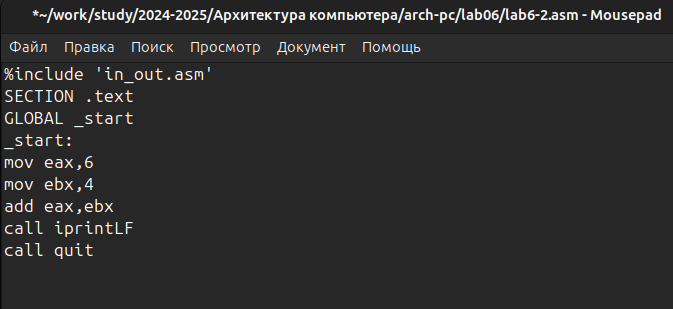


Рис. 10: Редактирование файла

Создаю и запускаю новый исполняемый файл.

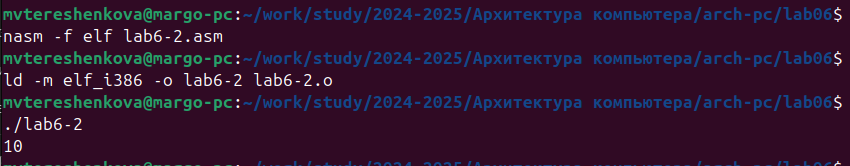


Рис. 11: Запуск исполняемого файла

Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

Заменяю в тексте программы функцию **iprintLF** на **iprint**.

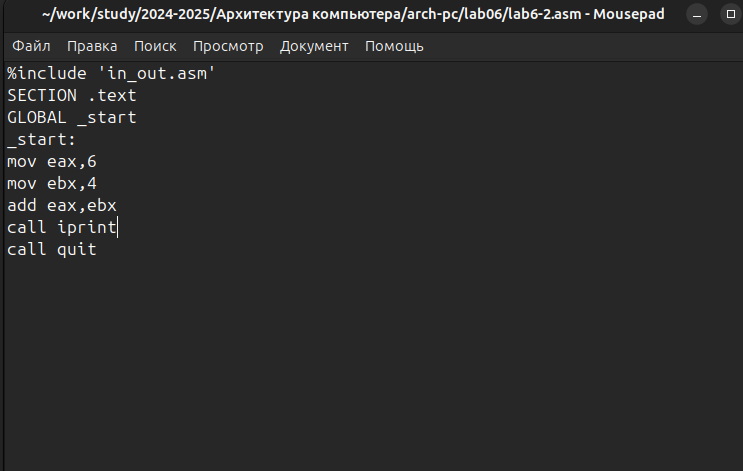


Рис. 12: Редактирование файла

Создаю и запускаю новый исполняемый файл.

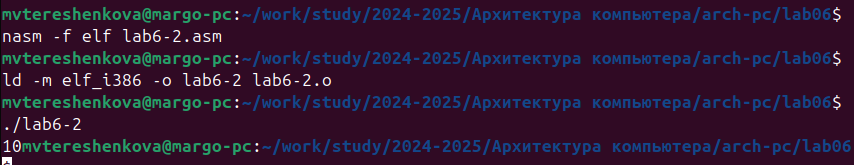


Рис. 13: Запуск исполняемого файла

**iprint** не добавляет к выводу символ переноса строки, в отличие от **iprintLF**.

## 4.2 Выполнение арифметических операций в NASM

1. Создаю файл lab6-3.asm с помощью утилиты **touch**.

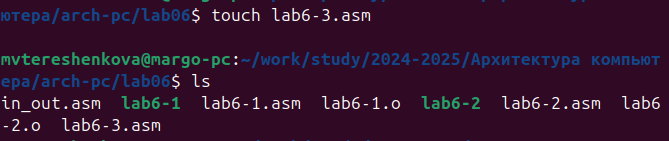


Рис. 14: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения *f(x)* = (5 \* 2 + 3)/3

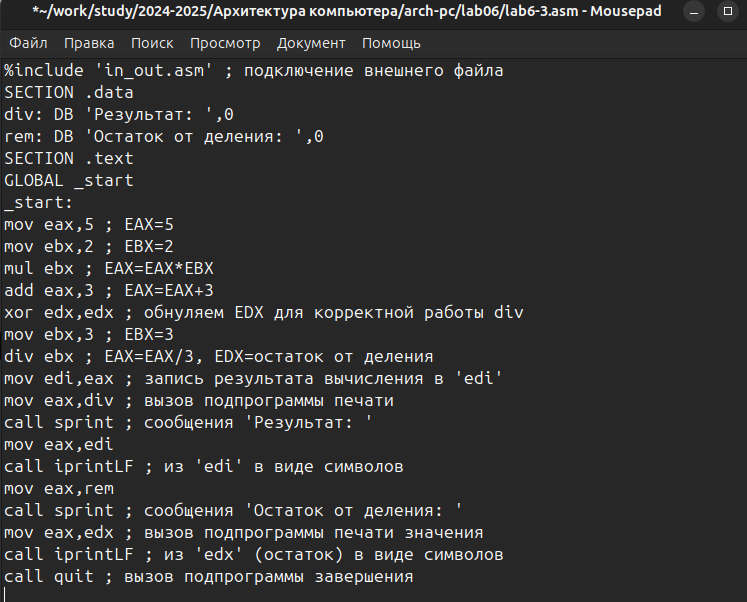


Рис. 15: Редактирование файла

Создаю исполняемый файл и запускаю его.

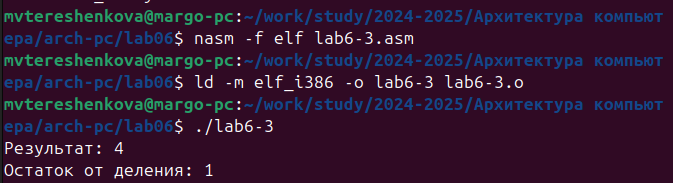


Рис. 16: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения *f(x)* = (4 \* 6 + 2)/5.

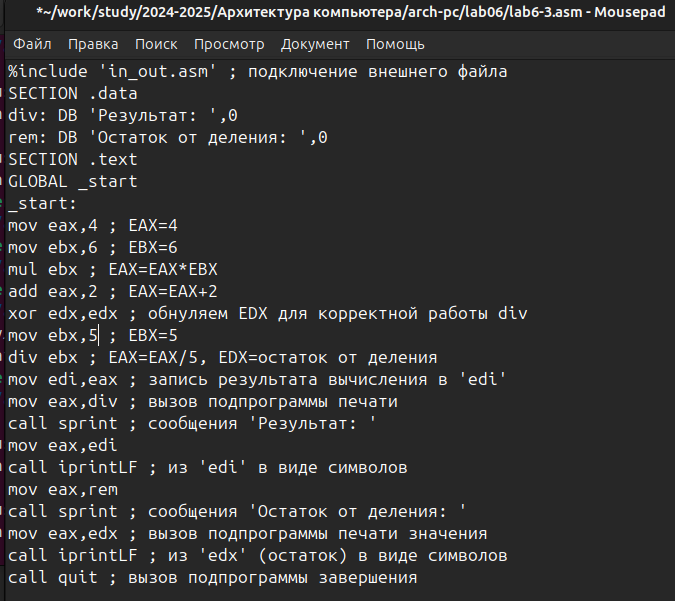


Рис. 17: Редактирование программы

Создаю и запускаю новый исполняемый файл.

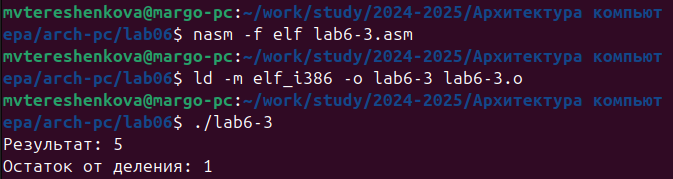


Рис. 18: Запуск исполняемого файла

Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

1. Создаю файл numbervariant.asm с помощью утилиты **touch**.

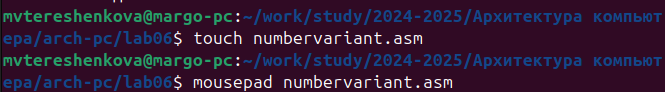


Рис. 19: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета.

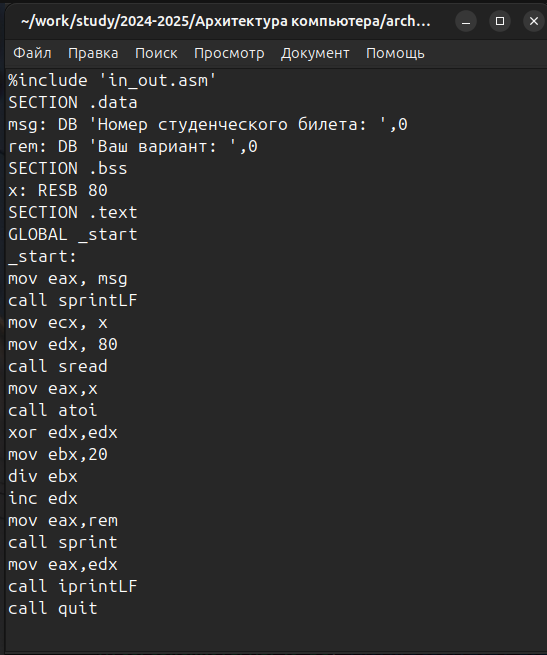


Рис. 20: Редактирование программы

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры.

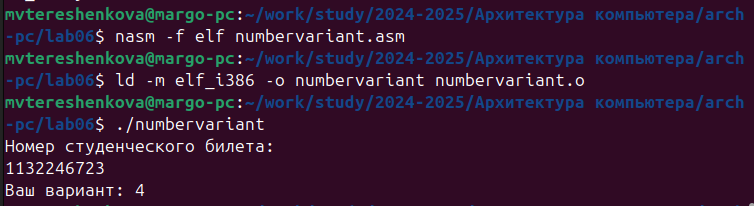


Рис. 21: Запуск исполняемого программы

Программа вывела, что мой вариант - 4.

## 4.3 Ответы на вопросы по программе.

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

**mov eax,rem**

**call sprint**

1. **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

1. **call atoi** используется для вызова подпрограммы из внешнего файла, которая преобразует ASCII-код символа в целое число и записывает результат в регистр eax.
2. 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*

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

**mov eax,edx**

**call iprintLF**

# 5 Выполнение самостоятельной работы

Создаю файл lab6-4.asm .

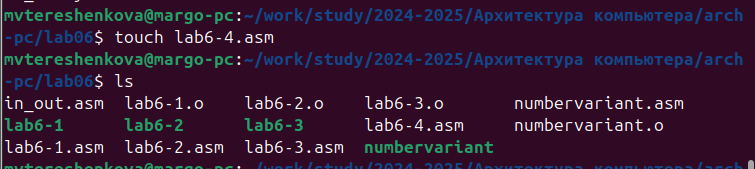


Рис. 22: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения 4/3\*(x-1)+5 . Это выражение было под вариантом 4.

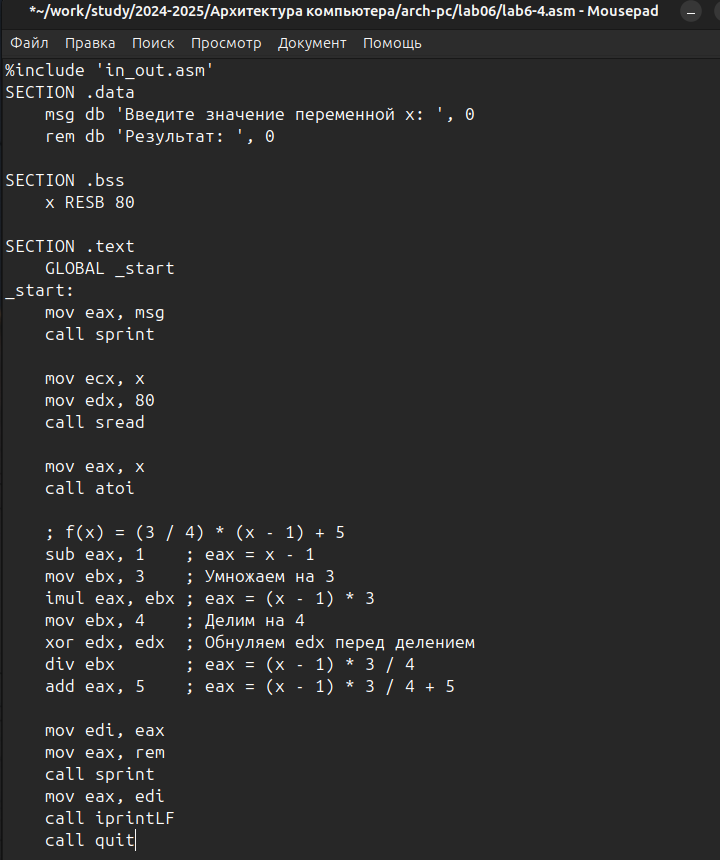


Рис. 23: Написание программы

Создаю и запускаю исполняемый файл 2 раза. Ввожу значение переменных x, которые написаны в моём варианте, а именно *x1=4* и *x2=10*.

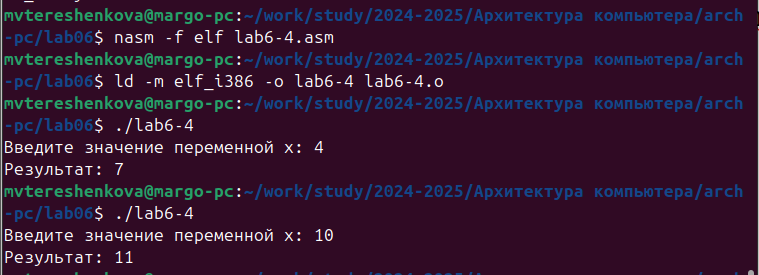


Рис. 24: Запуск исполняемого программы

Программа выводит верный ответ, учитывая, что при выполнении деления в качестве результата можно использовать только целую часть от деления и не учитывать остаток (т.е. 5 ∶ 2 = 2).

**Листинг** *Программа для вычисления значения выражения* 4/3\*(x-1)+5.

%include ‘in\_out.asm’

SECTION .data

msg db ‘Введите значение переменной x:’, 0

rem db ‘Результат:’, 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

; f(x) = (3 / 4) \* (x - 1) + 5

sub eax, 1 ; eax = x - 1

mov ebx, 3 ; Умножаем на 3

imul eax, ebx ; eax = (x - 1) \* 3

mov ebx, 4 ; Делим на 4

xor edx, edx ; Обнуляем edx перед делением

div ebx ; eax = (x - 1) \* 3 / 4

add eax, 5 ; eax = (x - 1) \* 3 / 4 + 5

mov edi, eax

mov eax, rem

call sprint

mov eax, edi

call iprintLF

call quit

# 6 Выводы

Благодаря данной лабораторной работе освоила арифметических инструкций языка ассемблера NASM.

# Список литературы

1. Лабораторная работа №6  
2. Таблица ASCII