Лабораторная работа №6

Дисциплина: Архитектура компьютера

Малюга Валерия Васильевна

Содержание

[1 Цель работы 1](#_Toc151123872)

[2 Задание 1](#_Toc151123873)

[3 Теоретическое введение 1](#_Toc151123874)

[4 Выполнение лабораторной работы 2](#_Toc151123875)

[4.1 Символьные и численные данные в NASM 2](#_Toc151123876)

[4.2 Выполнение арифметических операций в NASM 5](#_Toc151123877)

[4.2.1 Ответы на вопросы по программе 8](#_Toc151123878)

[4.3 Выполнение заданий для самостоятельной работы 9](#_Toc151123879)

[5 Выводы 11](#_Toc151123880)

[6 Список литературы 11](#_Toc151123881)

# 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

С помощью утилиты mkdir создала каталог для программам лабораторной работы № 6, перешла в него и создала файл lab6-1.asm Скопировала в текущий каталог файл in\_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах (рис. [1](#fig:001)).

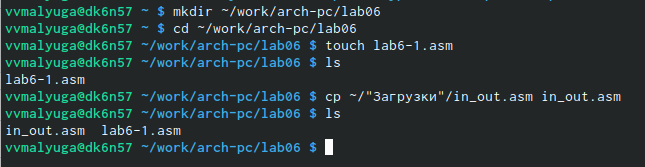


Рис. 1: Создание файла lab6-1.asm и копирование in\_out.asm

Ввела в файл lab6-1.asm текст программы из листинга 6.1 (рис. [2](#fig:002)).

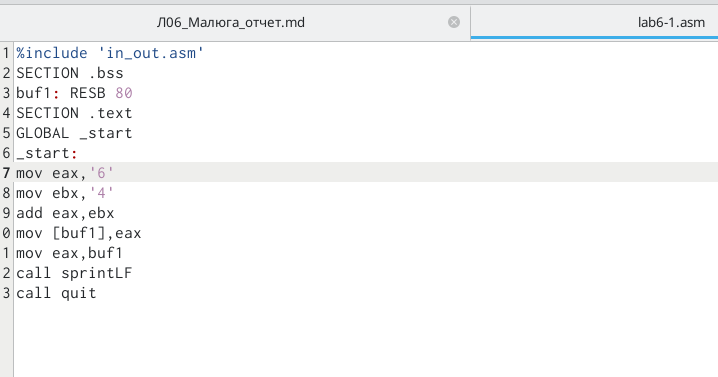


Рис. 2: Редактирование файла

Создала исполняемый файл и запустила его (рис. [3](#fig:003)). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

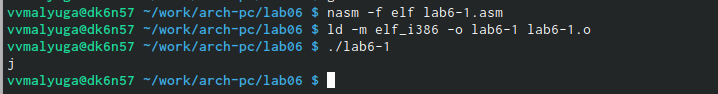


Рис. 3: Создание исполняемого файла и его запуск

Исправила текст программы: поменяла символы “6” и “4” на цифры 6 и 4 (рис. [4](#fig:004)). Создала исполняемый файл и запустила его (рис. [5](#fig:005)). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

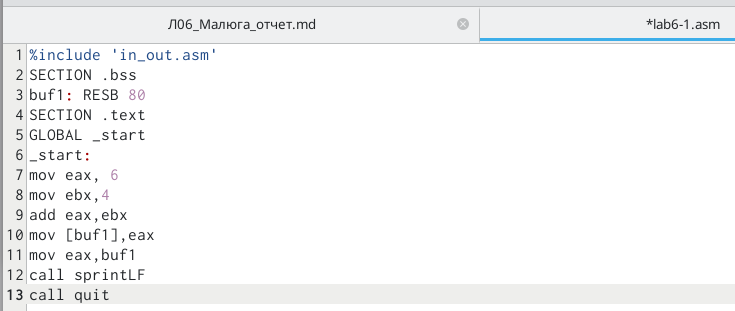


Рис. 4: Редактирование файла

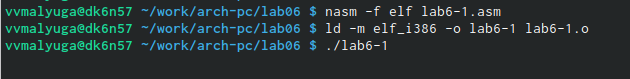


Рис. 5: Создание исполняемого файла и его запуск

Создала файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввела в него текст программы из листинга 6.2 (рис. [6](#fig:006)).

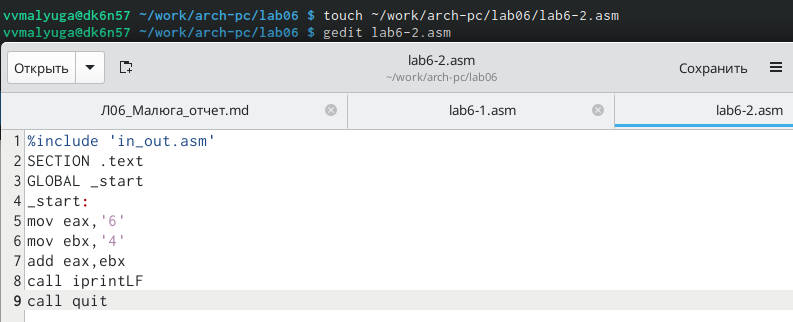


Рис. 6: Создание и редактирование файла lab6-2.asm

Создала исполняемый файл и запустила его (рис. [7](#fig:007)). Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.

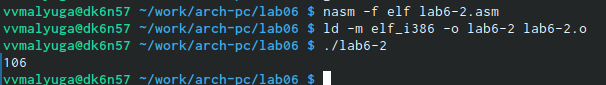


Рис. 7: Создание исполняемого файла и его запуск

Заменила в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. [8](#fig:008)).

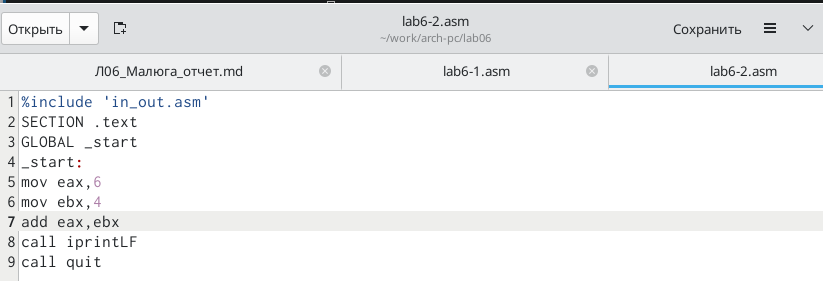


Рис. 8: Редактирование программы

Создала и запустила новый исполняемый файл (рис. [9](#fig:009)). Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

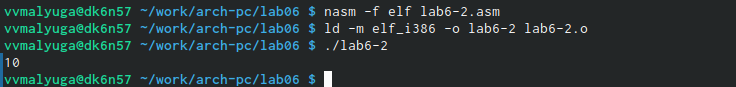


Рис. 9: Создание исполняемого файла и его запуск

Заменяю в тексте программы функцию iprintLF на iprint (рис. [10](#fig:010)). Создала и запустила новый исполняемый файл (рис. [10](#fig:010)). Вывод не изменился потому что символ переноса строки не отображался, когда программа исполнялась с функцией iprintLF, а iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF.

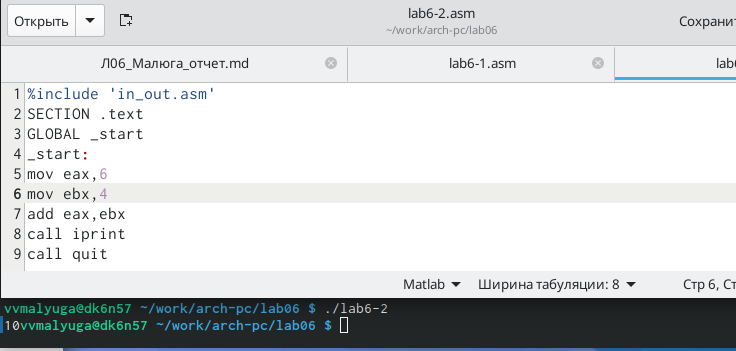


Рис. 10: Редактирование файла и его запуск

## 4.2 Выполнение арифметических операций в NASM

Создала файл lab6-3.asm с помощью утилиты touch (рис. [11](#fig:011)). Ввела в созданный файл текст программы для вычисления значения выражения f(x) = (5 \* 2 + 3)/3 (рис. [11](#fig:011)).

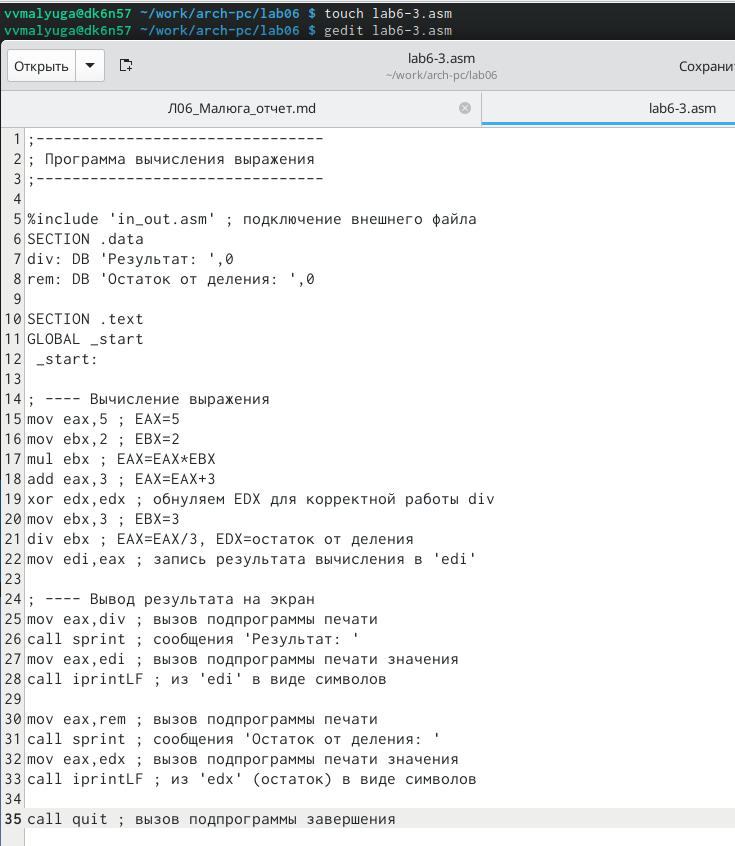


Рис. 11: Создание файла

Создала исполняемый файл и запустила его (рис. [[12](#fig:012)]).

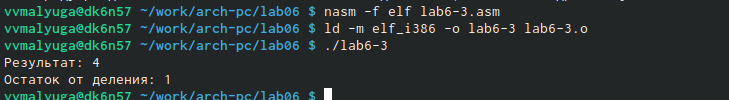


Рис. 12: Запуск исполняемого файла

Изменила программу так, чтобы она вычисляла значение выражения f(x) = (4 \* 6 + 2)/5 (рис. [13](#fig:013)). Создала и запустила новый исполняемый файл (рис. [13](#fig:013)). Программа отработала верно.

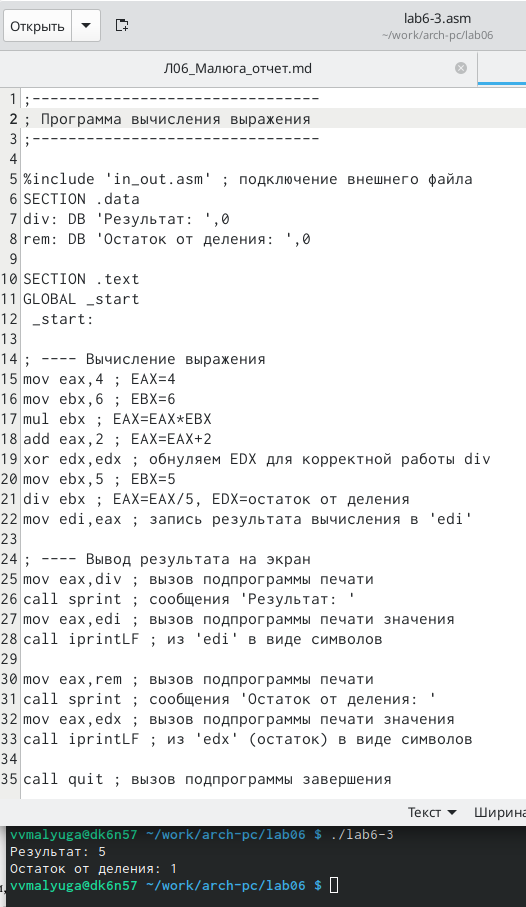


Рис. 13: Изменение программы и запуск исполняемого файла

Создала файл variant.asm с помощью утилиты touch. Ввела в файл текст из листинга 6.4. Создала исполняемый файл и запустила его. Ввела номер своего студенческого билета и получила номер своего варианта – 11 (рис. [14](#fig:014)). Проверила правильность выполнения, вычислив номер варианта аналитически.

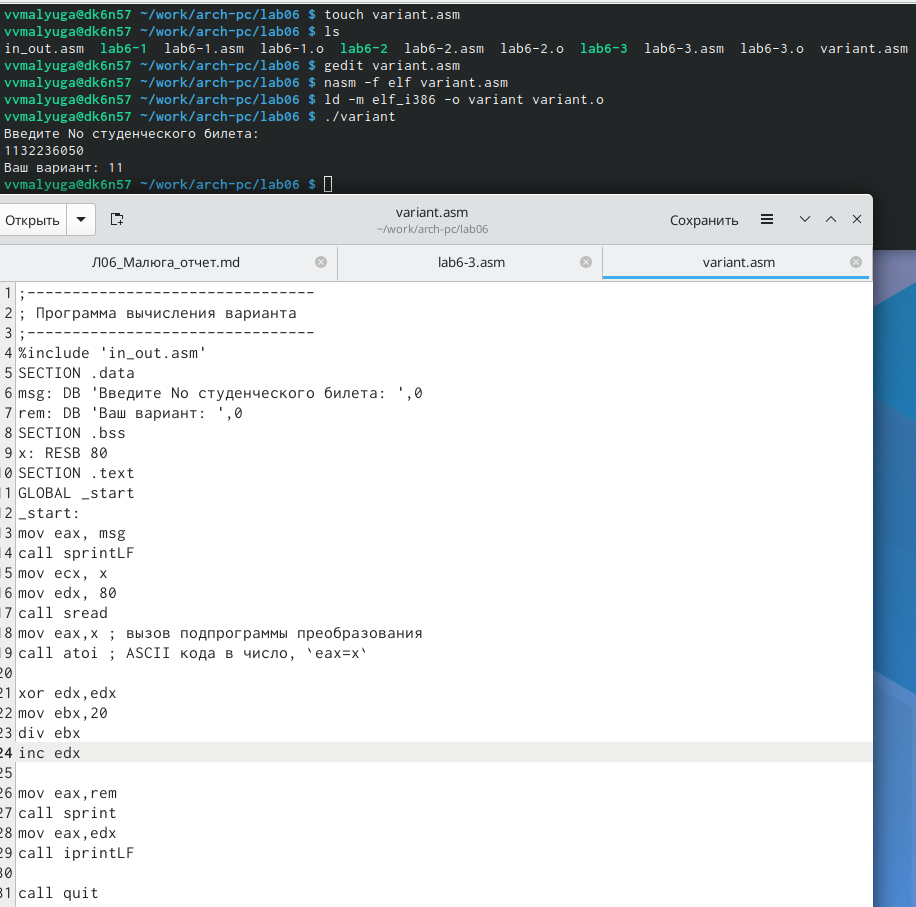


Рис. 14: Создание, редактирование и запуск файла

### 4.2.1 Ответы на вопросы по программе

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

mov eax,rem  
call sprint

1. 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
2. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax
3. 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

## 4.3 Выполнение заданий для самостоятельной работы

Создала файл zadanie.asm. Открыла его для редактирования, ввела в него текст программы для вычисления значения выражения 10\*(x + 1)-10 (вариант 11). Создала и запустила исполняемый файл. При вводе значения 1 на входе вывод программы = 10. При вводе значения 7 на входе вывод программы = 70. Значит, программа работает верно. (рис. [15](#fig:015)).

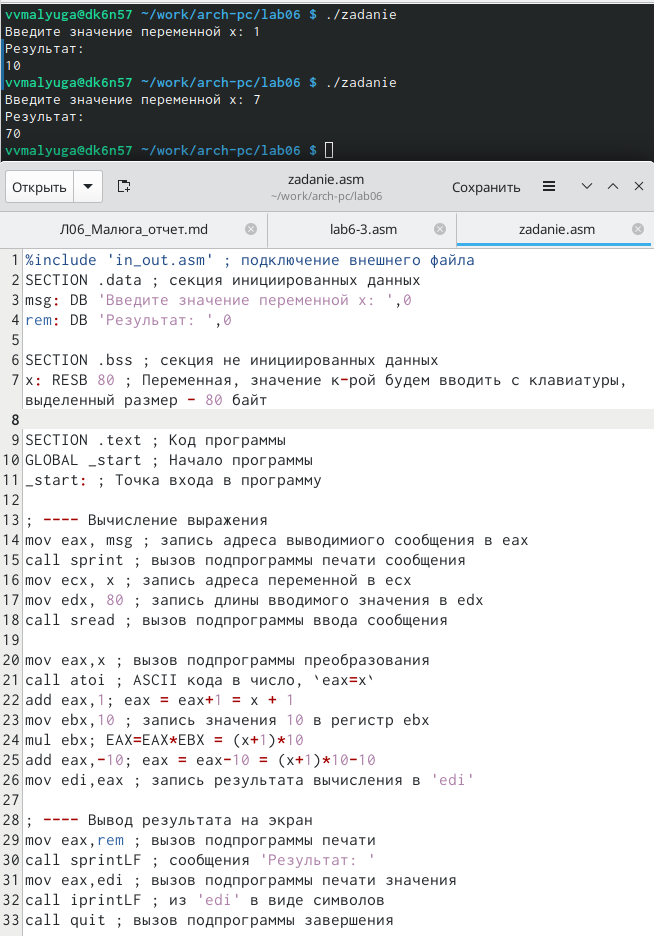


Рис. 15: Запуск программы для выполнения задания для самостоятельной работы

**Программа для вычисления значения выражения 10 \* (x + 1) - 10.**

%include 'in\_out.asm' ; подключение внешнего файла  
SECTION .data ; секция инициированных данных  
msg: DB 'Введите значение переменной х: ',0  
rem: DB 'Результат: ',0  
  
SECTION .bss ; секция не инициированных данных  
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный размер - 80 байт  
  
SECTION .text ; Код программы  
GLOBAL \_start ; Начало программы  
\_start: ; Точка входа в программу  
  
; ---- Вычисление выражения  
mov eax, msg ; запись адреса выводимиого сообщения в eax  
call sprint ; вызов подпрограммы печати сообщения  
mov ecx, x ; запись адреса переменной в ecx  
mov edx, 80 ; запись длины вводимого значения в edx  
call sread ; вызов подпрограммы ввода сообщения  
  
mov eax,x ; вызов подпрограммы преобразования  
call atoi ; ASCII кода в число, `eax=x`  
add eax,1; eax = eax+1 = x + 1  
mov ebx,10 ; запись значения 10 в регистр ebx  
mul ebx; EAX=EAX\*EBX = (x+1)\*10  
add eax,-10; eax = eax-10 = (x+1)\*10-10  
mov edi,eax ; запись результата вычисления в 'edi'  
  
; ---- Вывод результата на экран  
mov eax,rem ; вызов подпрограммы печати  
call sprintLF ; сообщения 'Результат: '  
mov eax,edi ; вызов подпрограммы печати значения  
call iprintLF ; из 'edi' в виде символов  
call quit ; вызов подпрограммы завершения

# 5 Выводы

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

# 6 Список литературы

1. [Лабораторная работа №7](https://esystem.rudn.ru/pluginfile.php/1584637/mod_resource/content/1/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%967.pdf)