Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютера

Зарицкая Марина Петровна

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

Создала каталог для программ лабораторной работы №6, перешла в него и создала файл lab6-1.asm (рис. [1](#fig:001)).

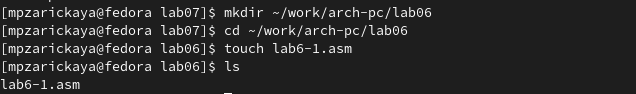


Figure 1: Создание каталога и файла программы

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

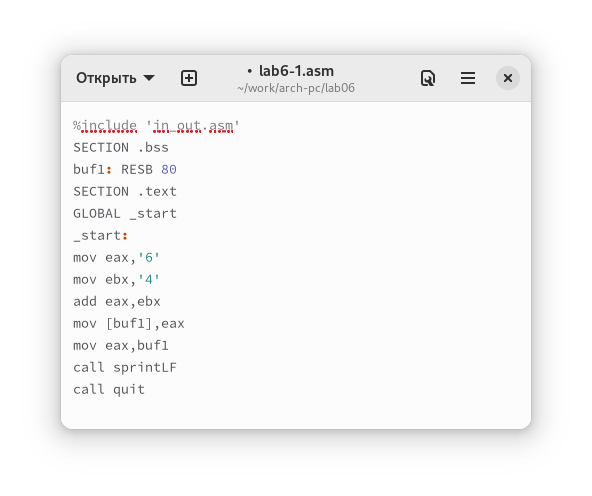


Figure 2: Программа из листинга 6.1

Создала исполняемый файл и запустила его. Программа вывела символ j (рис. [3](#fig:003)).

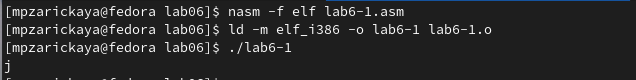


Figure 3: Результат работы программы

Изменила текст программы (Листинг 6.1) следующим образом: заменила строки

mov eax,'6'  
mov ebx,'4'

на строки

mov eax,6  
mov ebx,4

(рис. [4](#fig:004)).

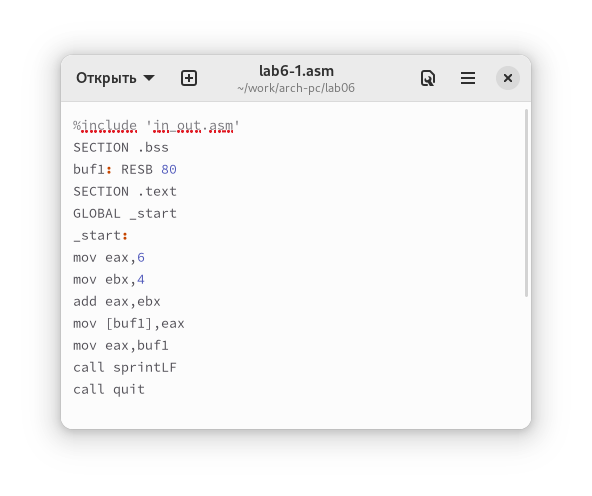


Figure 4: Измененный текст программы

Создала исполняемый файл и запустила его. Результат программы - символ с кодом 10, это символ перевода строки и он не отображается при выводе на экран (рис. [5](#fig:005)).

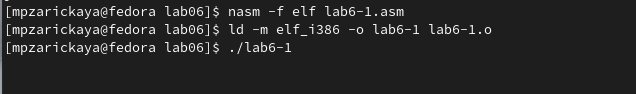


Figure 5: Результат работы измененной программы

Затем я 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)).



Figure 6: Текст программы из листинга 6.2

Запустила созданный исполняемый файл (рис. [7](#fig:007)).

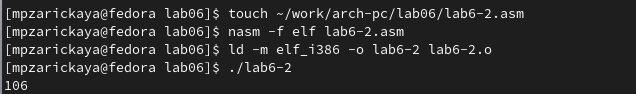


Figure 7: Запуск второй программы

В результате работы программы мы получим число 106. В данном случае, как и в первом, команда add складывает коды символов ‘6’ и ‘4’ (54+52=106). Однако, в отличии от программы из листинга 6.1, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущему примеру заменила строки программы

mov eax,'6'  
mov ebx,'4'

на строки

mov eax,6  
mov ebx,4

(рис. [8](#fig:008)).

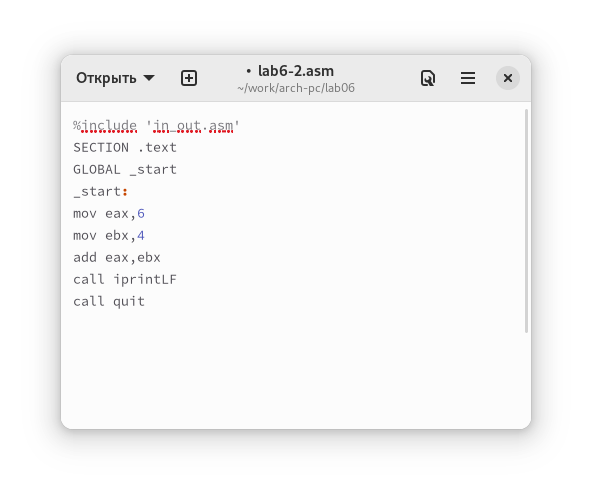


Figure 8: Измененный текст второй программы

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

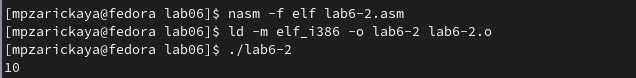


Figure 9: Результат работы программы

Изменила функцию iprintLF на iprint в тексте программы (рис. [10](#fig:010)).

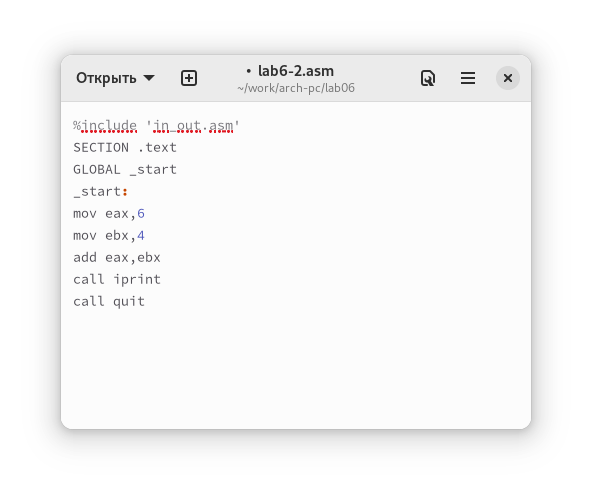


Figure 10: Изменения в тексте второй программы

Затем создала и запустила исполняемый файл. В результате работы программы так же выводится число 10, но уже без переноса строки, за который отвечает функция iprintLF (рис. [11](#fig:011)).

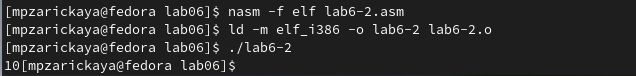


Figure 11: Вывод результата программы без переноса строки

## 4.2 Выполнение арифметических операций в NASM

Создала файл lab6-3.asm в каталоге ~/work/arch-pc/lab06 (рис. [12](#fig:012)).

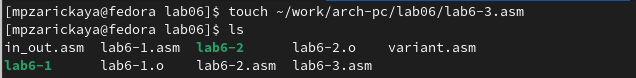


Figure 12: Создание файла третьей программы

Ввела текст программы из листинга 6.3 в файл lab6-3.asm (рис. [13](#fig:013)).

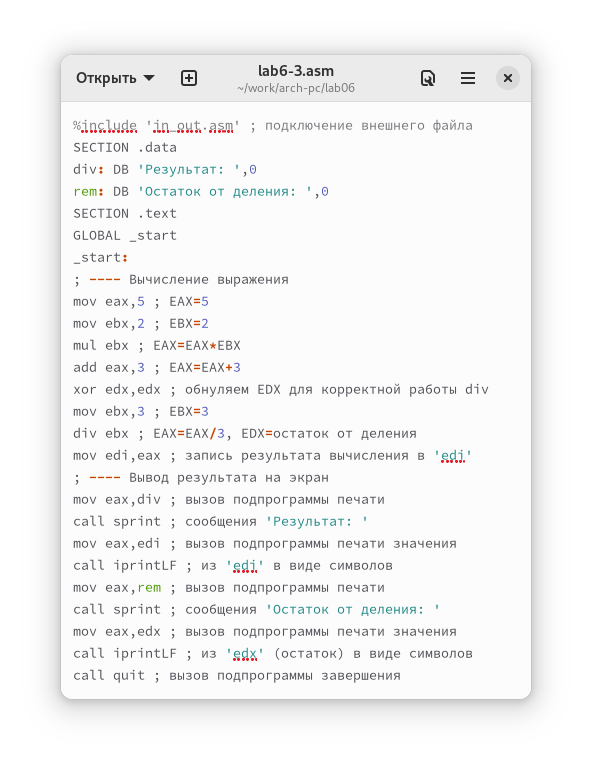


Figure 13: Текст программы из листинга 6.3

Создала и запустила исполняемый файл (рис. [14](#fig:014)).

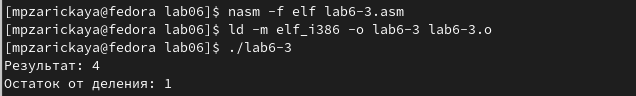


Figure 14: Результат работы третьей программы

Изменила программу так, чтобы она вычисляла значение выражения f(x) = (4 \* 6 + 2)/5 (рис. [15](#fig:015)).

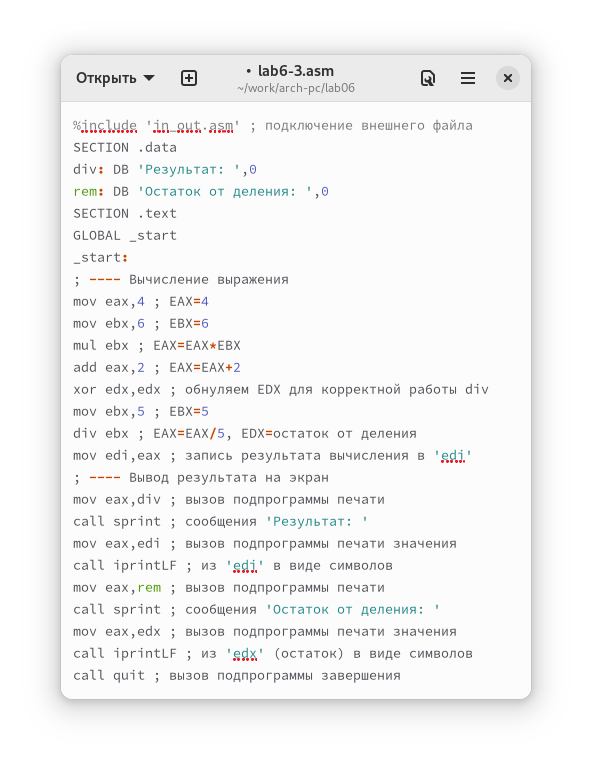


Figure 15: Измененный текст программы

Создала и запустила новый исполняемый файл (рис. [16](#fig:016)).

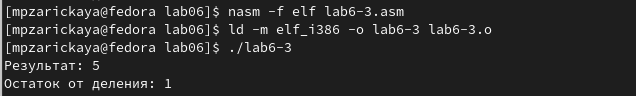


Figure 16: Вывод измененной программы

Создала файл variant.asm с помощью утилиты touch (рис. [17](#fig:017)).

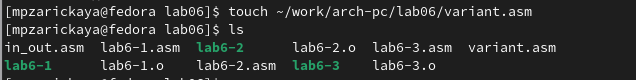


Figure 17: Создание файла

Ввела в файл текст программы из листинга 6.4 (рис. [18](#fig:018)).

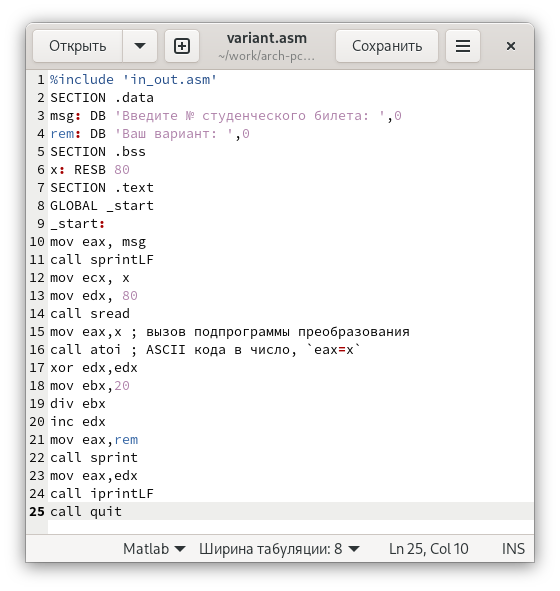


Figure 18: Текст программы из листинга 6.4

Создала исполняемый файл и запустила программу. Ввела номер своего студенческого билета и получила номер своего варианта - 7 (рис. [19](#fig:019)). Проверила правильность выполнения, вычислив номер варианта аналитически.

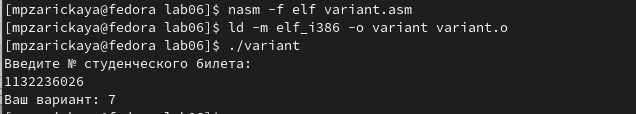


Figure 19: Результат работы программы для вычисления варианта

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 5(x − 1)^2 в соответствии со своим вариантом (рис. [20](#fig:020)).

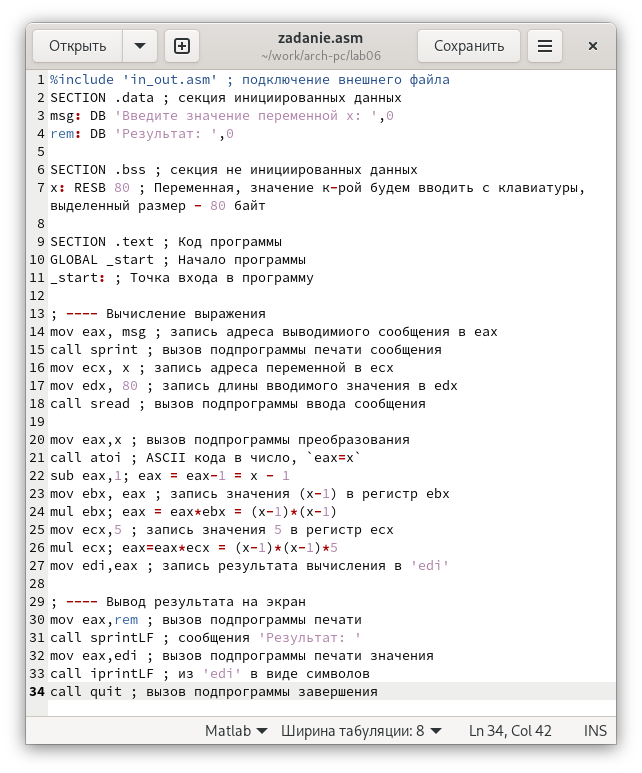


Figure 20: Текст программы для вычисления значения выражения

Создала и запустила исполняемый файл. При вводе значения 3 на входе вывод программы - 20. При вводе значения 5 программа выводит результат 80. Программа работает верно (рис. [21](#fig:021)).

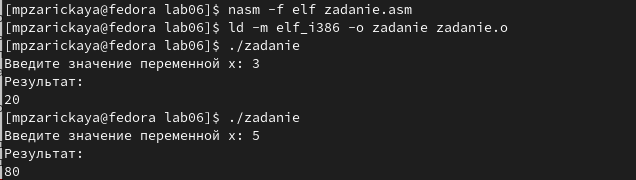


Figure 21: Результат работы программы для вычисления значения выражения

**Листинг программы для вычисления значения выражения 5(x − 1)^2 (вариант 7)**

%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`  
sub eax,1; eax = eax-1 = x - 1  
mov ebx, eax ; запись значения (x-1) в регистр ebx   
mul ebx; eax = eax\*ebx = (x-1)\*(x-1)  
mov ecx,5 ; запись значения 5 в регистр ecx  
mul ecx; eax=eax\*ecx = (x-1)\*(x-1)\*5  
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.

# Список литературы

(https://esystem.rudn.ru/mod/resource/view.php?id=1030554)