Отчёт по лабораторной работе 6

Дисциплина: Архитектура Компьютера

Ислам Карданов Вячеславович

Содержание

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

# 2 Задание

1. Изучение типов данных в ассемблере
2. Изучение арифметических операций в ассемблере
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. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. Существует три основных способа адресации:

1. Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: mov ax,bx.
2. Непосредственная адресация – значение операнда задается непосредственно в команде, Например: mov ax,2.
3. Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

* Схема команды целочисленного сложения add (от англ. addition - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда.
* Команда целочисленного вычитания sub (от англ. subtraction – вычитание) работает аналогично команде add.
* Существуют специальные команды: inc (от англ. increment) и dec (от англ. decrement), которые увеличивают и уменьшают на 1 свой операнд.
* Умножение и деление, в отличии от сложения и вычитания, для знаковых и беззнаковых чисел производиться по-разному, поэтому существуют различные команды.
* Для беззнакового умножения используется команда mul (от англ. multiply – умножение), для знакового умножения используется команда imul.
* Для деления, как и для умножения, существует 2 команды div (от англ. divide - деление) и idiv.

# 4 Выполнение лабораторной работы

## 4.1 Символьные и численные данные в NASM

Я создаю каталог для программ лабораторной работы № 6, перехожу в него и создаю файл lab6-1.asm.

Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистр eax.

В данной программе (рис. [[1](#fig:001)]) в регистр eax записывается символ 6 (mov eax,‘6’), в регистр ebx символ 4 (mov ebx,‘4’). Далее к значению в регистре eax прибавляем значение регистра ebx (add eax,ebx, результат сложения запишется в регистр eax). Далее выводим результат. Так как для работы функции sprintLF в регистр eax должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра eax в переменную buf1 (mov [buf1],eax), а затем запишем адрес переменной buf1 в регистр eax (mov eax,buf1) и вызовем функцию sprintLF.



Figure 1: Редактирую файл lab6-1.asm

В данном случае при выводе значения регистра eax мы ожидаем увидеть число 10. Однако результатом будет символ j. (рис. [[2](#fig:002)])

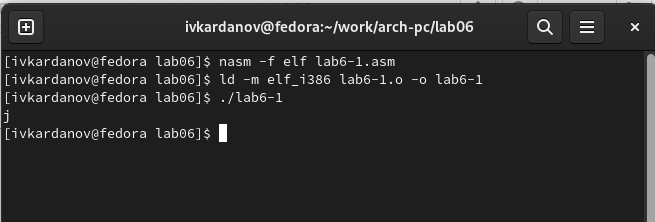


Figure 2: Запуск файла lab6-1.asm

Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа 4 – 00110100 (52). Команда add eax,ebx запишет в регистр eax сумму кодов – 01101010 (106), что в свою очередь является кодом символа j.

Далее изменяю текст программы и вместо символов, запишем в регистры числа. (рис. [[3](#fig:003)])

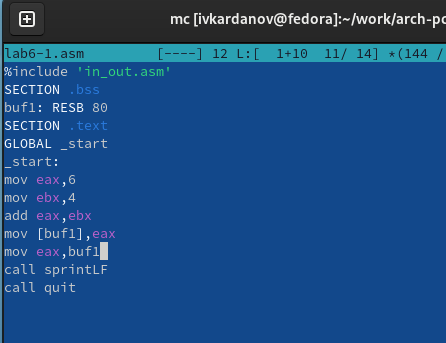


Figure 3: Редактирую файл lab6-1.asm

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. (рис. [[4](#fig:004)]) Это символ конца строки (возврат каретки). В консоле он не отображается, но добавляет пустую строку.

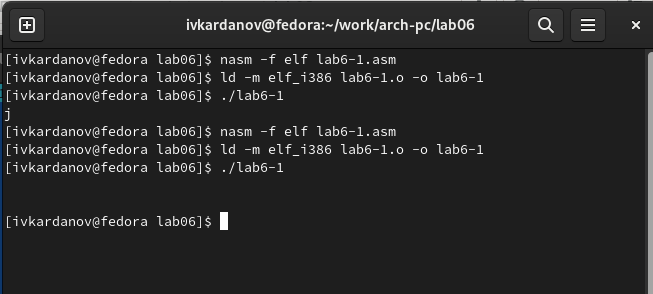


Figure 4: Запуск файла lab6-1.asm

Как отмечалось выше, для работы с числами в файле in\_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразовал текст программы с использованием этих функций. (рис. [[5](#fig:005)])

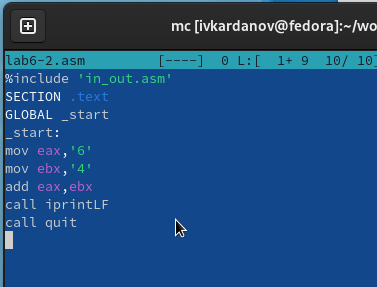


Figure 5: Редактирую файл lab6-2.asm

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). (рис. [[6](#fig:006)]) Однако, в отличии от прошлой программы, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

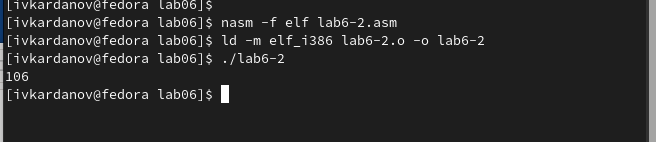


Figure 6: Запуск файла lab6-2.asm

Аналогично предыдущему примеру изменим символы на числа. (рис. [[7](#fig:007)])

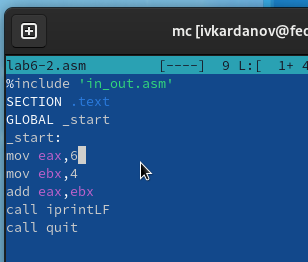


Figure 7: Редактирую файл lab6-2.asm

Функция iprintLF позволяет вывести число и операндами были числа (а не коды символов). Поэтому получаем число 10. (рис. [[8](#fig:008)])

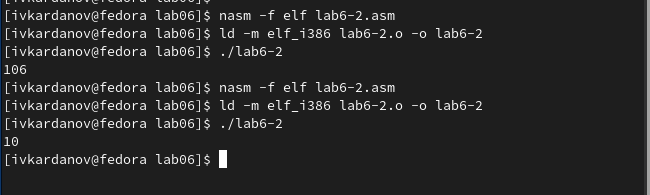


Figure 8: Запуск файла lab6-2.asm

Заменил функцию iprintLF на iprint. Создал исполняемый файл и запустил его. Вывод отличается тем, что нет переноса строки.(рис. [[9](#fig:009)])

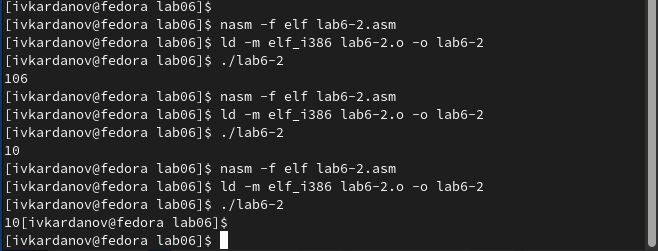


Figure 9: Запуск файла lab6-2.asm

## 4.2 Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения . (рис. [[10](#fig:010)] [[11](#fig:011)])

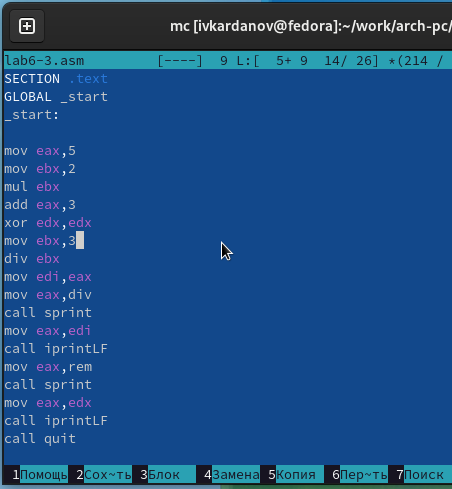


Figure 10: Редактирую файл lab6-3.asm

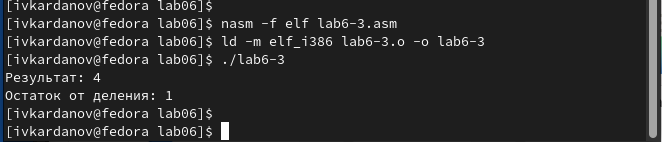


Figure 11: Запуск файла lab6-3.asm

Изменил текст программы для вычисления выражения . Создал исполняемый файл и проверил его работу. (рис. [[12](#fig:012)] [[13](#fig:013)])



Figure 12: Редактирую файл lab6-3.asm

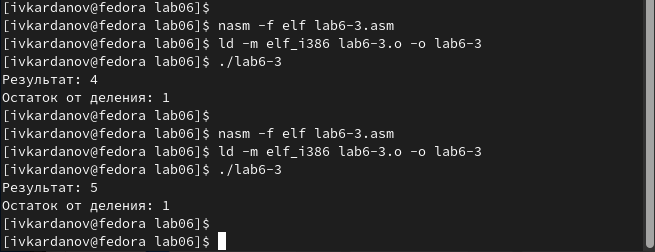


Figure 13: Запуск файла lab6-3.asm

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета. (рис. [[14](#fig:014)])

В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. Как отмечалось выше ввод с клавиатуры осуществляется в символьном виде и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого может быть использована функция atoi из файла in\_out.asm. (рис. [[15](#fig:015)])

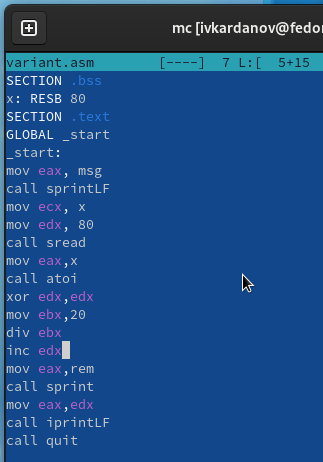


Figure 14: Редактирую файл variant.asm

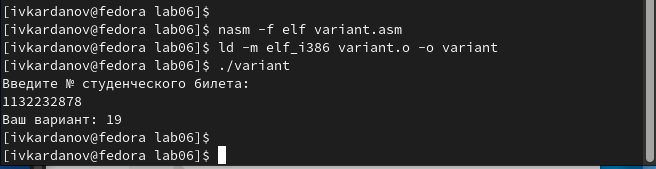


Figure 15: Запуск файла variant.asm

## 4.3 ответы на вопросы по программе variant.asm

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

Перекладывает значение переменной с фразой ‘Ваш вариант:’ в регистр eax: mov eax, rem

Вызывает подпрограмму вывода строки: call sprint

1. Для чего используется следующие инструкции?

mov ecx, x   
mov edx, 80   
call sread

Считывают значение студентского билета в переменную X из консоли.

1. Для чего используется инструкция “call atoi”?

Инструкция “call atoi” используется для преобразования введенных символов в числовой формат.

1. Какие строки листинга отвечают за вычисления варианта?

xor edx,edx  
mov ebx,20  
div ebx  
inc edx

Выполняется деление номера студенческого билета на 20 и остаток сохраняется в регистре edx. Затем к остатку прибавляется 1.

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

Остаток от деления записывается в регистр edx.

1. Для чего используется инструкция “inc edx”?

Инструкция “inc edx” используется для увеличения значения в регистре edx на 1. В данном случае, она используется для добавления единицы к остатку от деления.

1. Какие строки листинга отвечают за вывод на экран результата вычислений?

Результат вычислений перекладывается в регистр eax: mov eax, edx

Вызывается подпрограмма вывода строки: call iprintLF

## 4.4 Задание для самостоятельной работы

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 x1 и x2 из 6.3. (рис. [[16](#fig:016)] [[17](#fig:017)])

Получили вариант 19 - для

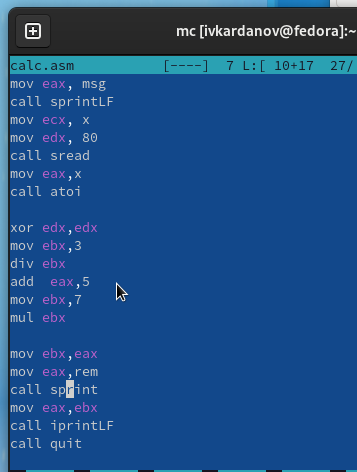


Figure 16: Редактирую файл calc.asm

Если подставить получается .

Если подставить получается .

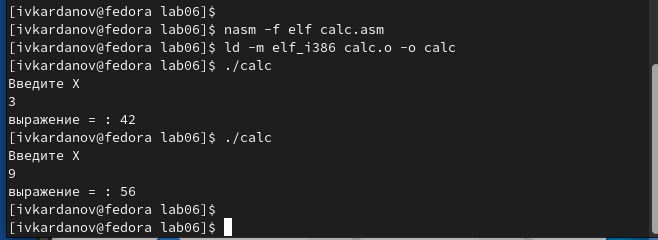


Figure 17: Запуск файла calc.asm

Программа считает верно.

# 5 Выводы

Изучили работу с арифметическими операциями.

# Список литературы

1. [Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/2089086/mod_resource/content/0/Лабораторная%20работа%20№6.%20Арифметические%20операции%20в%20NASM..pdf)