Отчёт по лабораторной работе 5

Дисциплина: архитектура компьютера

Плугин Никита

Содержание

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

# 2 Задания

1. Изучить инструкции ассемблера для арифметических действий
2. Написать программы по заданиям
3. Узнать свой вариант
4. Выполнить самостоятельное задание

# 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:

* 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.

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

1. Создал каталог для программам лабораторной работы № 6, перешел в него и создал файл lab6-1.asm. (рис. [[1](#fig:001)])

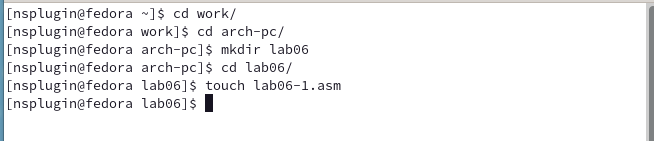


Figure 1: Подготовил каталог

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



Figure 2: Программа в файле lab6-1.asm

В данной программе (рис. [[2](#fig:002)]) мы записываем символ ‘6’ в регистр eax (mov eax, ‘6’), а символ ‘4’ в регистр ebx (mov ebx, ‘4’). Затем мы добавляем значение регистра ebx к значению в регистре eax (add eax, ebx, результат сложения записывается в регистр eax). После этого мы выводим результат. Однако, для использования функции sprintLF, необходимо, чтобы в регистре eax был записан адрес, поэтому мы используем дополнительную переменную. Мы записываем значение регистра eax в переменную buf1 (mov [buf1], eax), а затем записываем адрес переменной buf1 в регистр eax (mov eax, buf1) и вызываем функцию sprintLF.

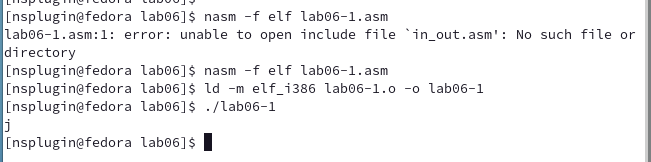


Figure 3: Запуск программы lab6-1.asm

В данном случае, когда мы ожидаем увидеть число 10 при выводе значения регистра eax, фактическим результатом будет символ ‘j’. Это происходит из-за того, что код символа ‘6’ равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа ‘4’ равен 00110100 (или 52 в десятичном представлении). Когда мы выполняем команду add eax, ebx, результатом будет сумма кодов - 01101010 (или 106 в десятичном представлении), который соответствует символу ‘j’. (рис. [[3](#fig:003)])

1. Далее изменяю текст программы и вместо символов, запишем в регистры числа. (рис. [[4](#fig:004)])

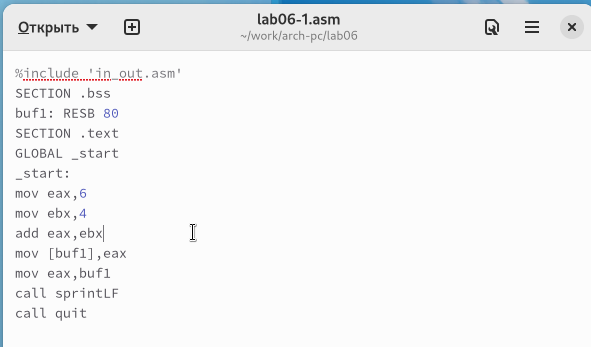


Figure 4: Программа в файле lab6-1.asm

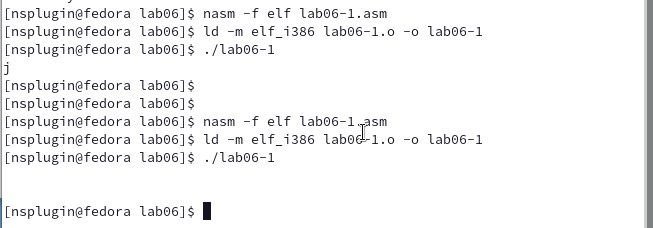


Figure 5: Запуск программы lab6-1.asm

Как и в предыдущем случае, при выполнении программы мы не получим число 10. Вместо этого выводится символ с кодом 10, который представляет собой символ конца строки (возврат каретки). (рис. [[5](#fig:005)]) Этот символ не отображается в консоли, но он добавляет пустую строку.

1. 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. [[6](#fig:006)])

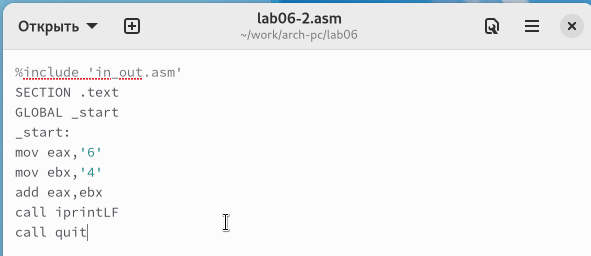


Figure 6: Программа в файле lab6-2.asm

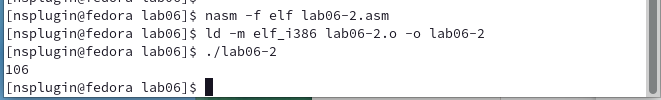


Figure 7: Запуск программы lab6-2.asm

В результате выполнения программы мы получим число 106. (рис. [[7](#fig:007)]) В данном случае, как и в первом случае, команда add складывает коды символов ‘6’ и ‘4’ (54+52=106). Однако, в отличие от предыдущей программы, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

1. Аналогично предыдущему примеру изменим символы на числа.(рис. [[8](#fig:008)])



Figure 8: Программа в файле lab6-2.asm

Функция iprintLF позволяет вывести число и операндами были числа (а не коды символов). Поэтому получаем число 10.(рис. [[9](#fig:009)])

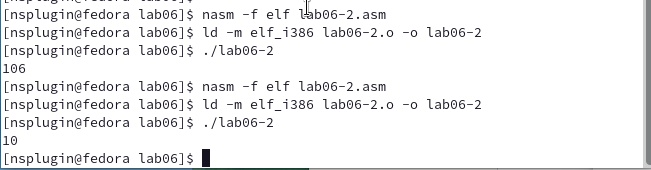


Figure 9: Запуск программы lab6-2.asm

Заменил функцию iprintLF на iprint. Создал исполняемый файл и запустил его. Вывод отличается тем, что нет переноса строки.(рис. [[10](#fig:010)])

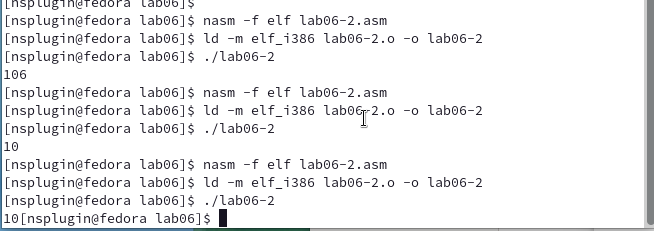


Figure 10: Запуск программы lab6-2.asm

## 4.2 Выполнение арифметических операций в NASM

1. В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения (рис. [[11](#fig:011)]) (рис. [[12](#fig:012)])

* .

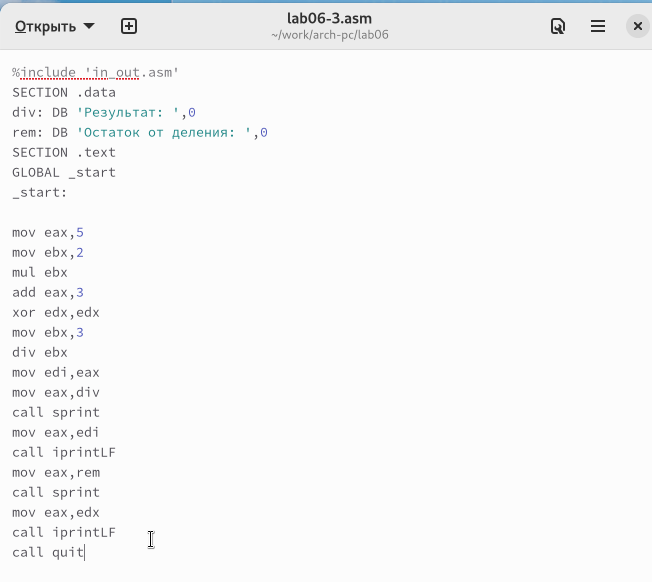


Figure 11: Программа в файле lab6-3.asm

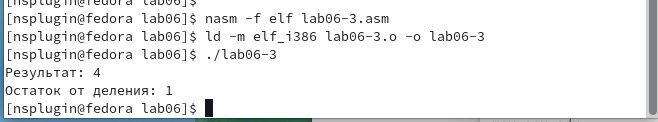


Figure 12: Запуск программы lab6-3.asm

Изменил текст программы для вычисления выражения

. Создал исполняемый файл и проверил его работу. (рис. [[13](#fig:013)]) (рис. [[14](#fig:014)])

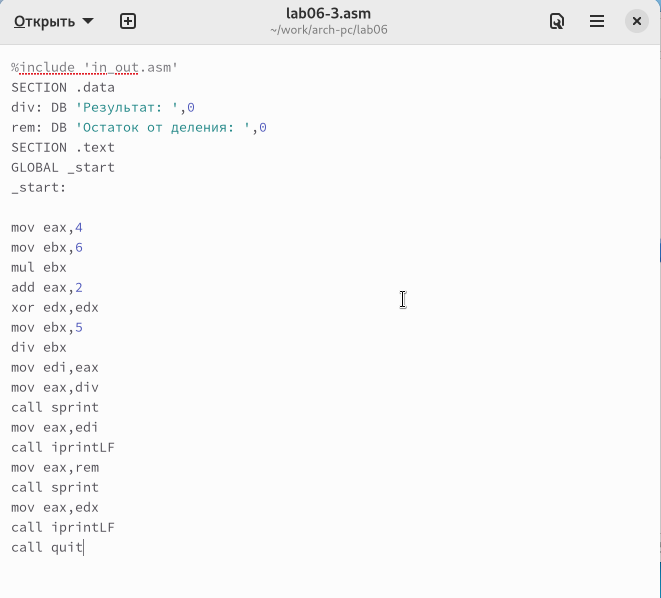


Figure 13: Программа в файле lab6-3.asm

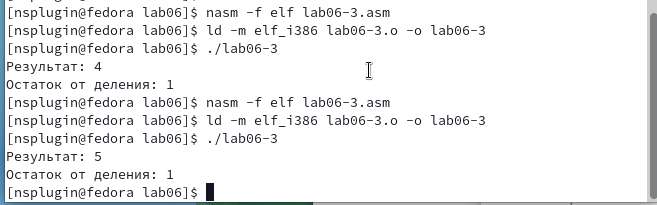


Figure 14: Запуск программы lab6-3.asm

1. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета. (рис. [[15](#fig:015)]) (рис. [[16](#fig:016)])

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.

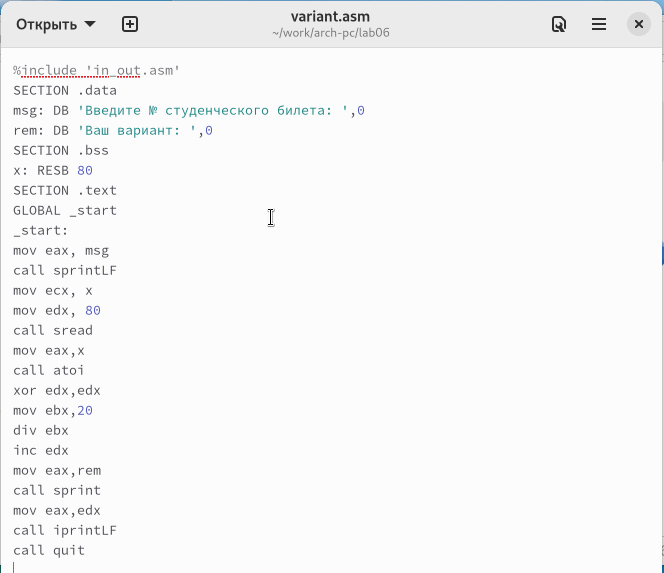


Figure 15: Программа в файле variant.asm

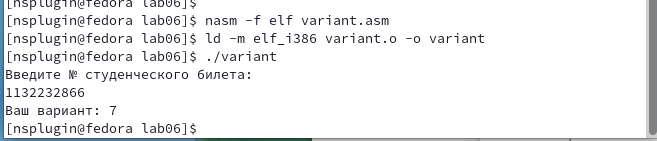


Figure 16: Запуск программы variant.asm

### 4.2.1 Ответы на вопросы по программе variant.asm

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

Строка “mov eax, rem” перекладывает в регистр значение переменной с фразой “Ваш вариант:”

Строка “call sprint” вызывает подпрограмму вывода строки

1. Для чего используются следующие инструкции?

Инструкция “nasm” используется для компиляции кода на языке ассемблера NASM

Инструкция “mov ecx, x” используется для перемещения значения переменной x в регистр ecx

Инструкция “mov edx, 80” используется для перемещения значения 80 в регистр edx

Инструкция “call sread” вызывает подпрограмму для считывания значения студенческого билета 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?

Строка “xor edx, edx” обнуляет регистр edx

Строка “mov ebx, 20” записывает значение 20 в регистр ebx

Строка “div ebx” выполняет деление номера студенческого билета на 20

Строка “inc edx” увеличивает значение регистра edx на 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 соответствии с формулой вычисления варианта

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

Строка “mov eax, edx” перекладывает результат вычислений в регистр eax

Строка “call iprintLF” вызывает подпрограмму для вывода значения на экран

## 4.3 Самостоятельное 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 x1 и x2 из 6.3.

Получили вариант 7 -

для

(рис. [[17](#fig:017)]) (рис. [[18](#fig:018)])

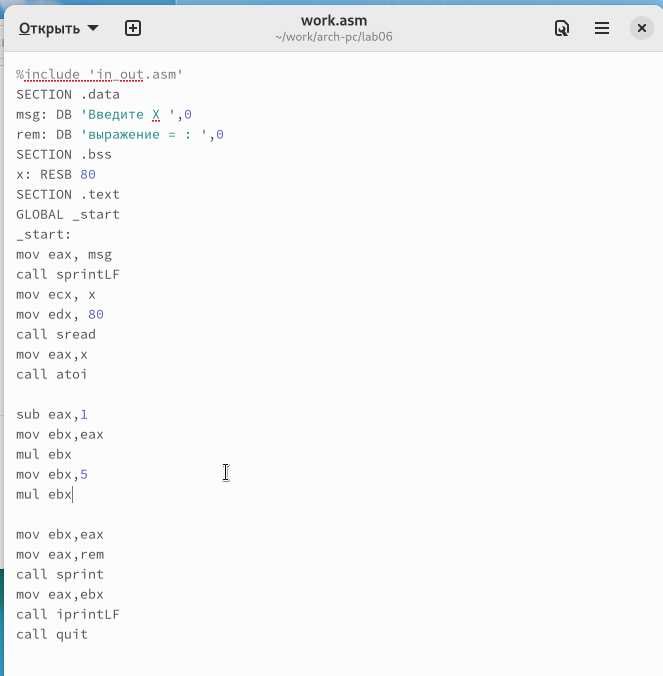


Figure 17: Программа в файле work.asm

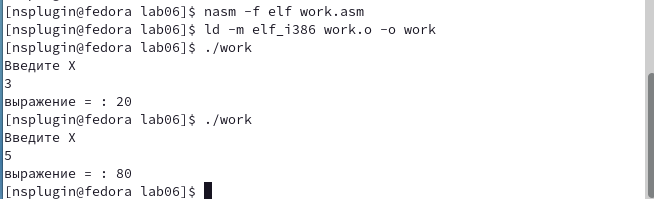


Figure 18: Запуск программы work.asm

# 5 Выводы

Изучили работу с арифметическими операциями.