Отчёт по лабораторной работе 6

Архитектура компьютеров

Махкамов Рауфджон НММбд-04-24

Содержание

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

# 2 Теоретическое введение

В ассемблере можно выделить следующие базовые операции:

* 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 отличие от сложения и вычитания, для знаковых и беззнаковых чисел производятся 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, существуют две команды: div (от англ. **divide** - деление) и idiv.

# 3 Выполнение лабораторной работы

## 3.1 Символьные и численные данные в NASM

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 в регистр eax записывается символ 6 (используя команду mov eax, '6'), в регистр ebx записывается символ 4 (используя команду mov ebx, '4'). Далее к значению в регистре eax прибавляем значение регистра ebx (командой add eax, ebx, результат сложения запишется в регистр eax). После этого выводим результат. (изображение 1) (изображение 2)

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 (командой mov [buf1], eax), а затем запишем адрес переменной buf1 в регистр eax (командой mov eax, buf1) и вызовем функцию sprintLF.

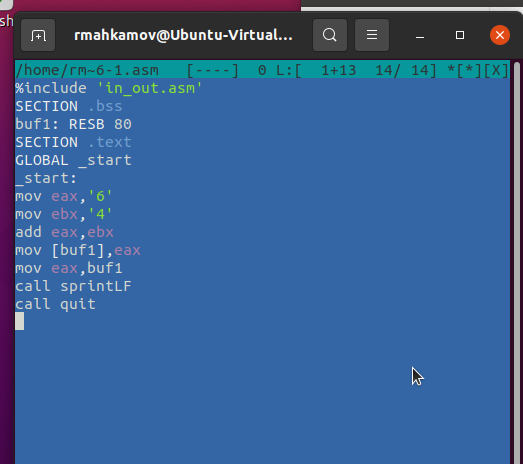


Рис. 1: Программа lab6-1.asm

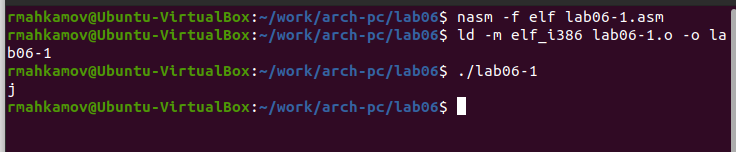


Рис. 2: Запуск программы lab6-1.asm

В данном случае при выводе значения регистра eax мы ожидаем увидеть число 10. Однако результатом будет символ j. 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. (изображение 3) (изображение 4)

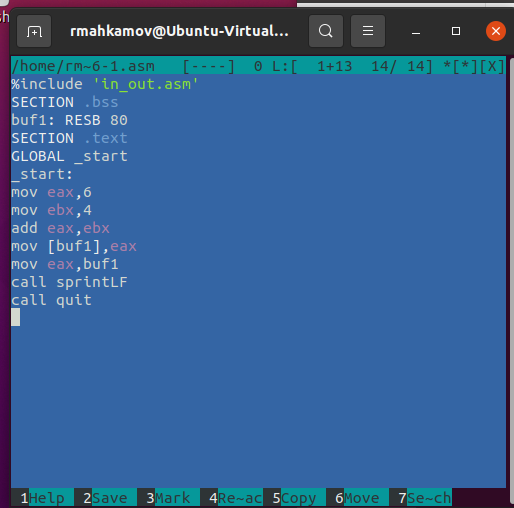


Рис. 3: Программа lab6-1.asm с числами

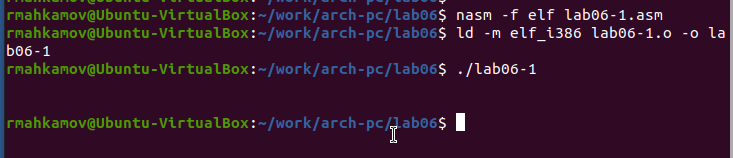


Рис. 4: Запуск программы lab6-1.asm с числами

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Это символ конца строки (возврат каретки). В консоли он не отображается, но добавляет пустую строку.

Как отмечалось выше, для работы с числами в файле in\_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразую текст программы с использованием этих функций. (изображение 5) (изображение 6)

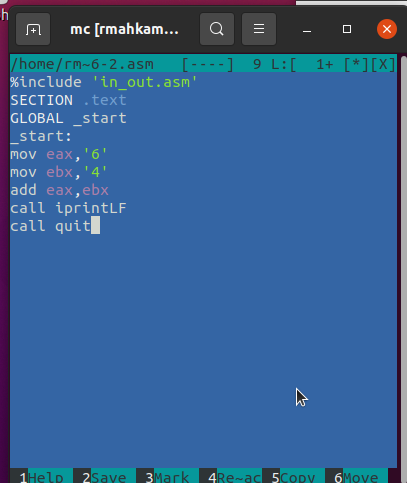


Рис. 5: Программа lab6-2.asm

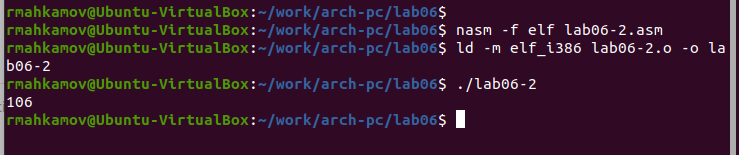


Рис. 6: Запуск программы lab6-2.asm

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). Однако, в отличие от прошлой программы, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущему примеру изменим символы на числа. (изображение 7) (изображение 8)

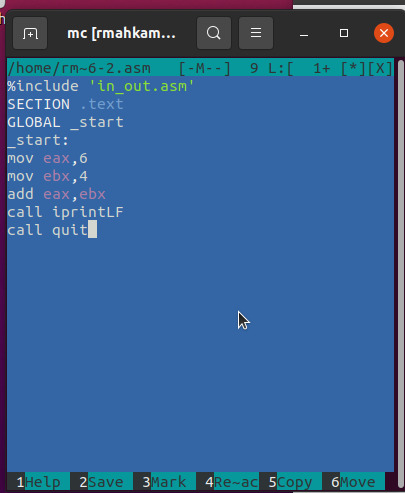


Рис. 7: Программа lab6-2.asm с числами

Функция iprintLF позволяет вывести число, и операндами были числа (а не коды символов). Поэтому получаем число 10.

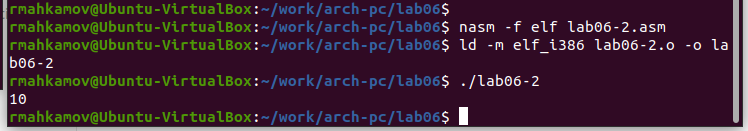


Рис. 8: Запуск программы lab6-2.asm с числами

Заменил функцию iprintLF на iprint. Создал исполняемый файл и запустил его. Вывод отличается тем, что нет переноса строки. (изображение 9)

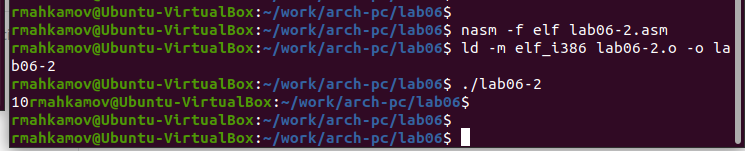


Рис. 9: Запуск программы lab6-2.asm без переноса строки

## 3.2 Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $ f(x) = (5 \* 2 + 3)/3 $. (изображение 10) (изображение 11)

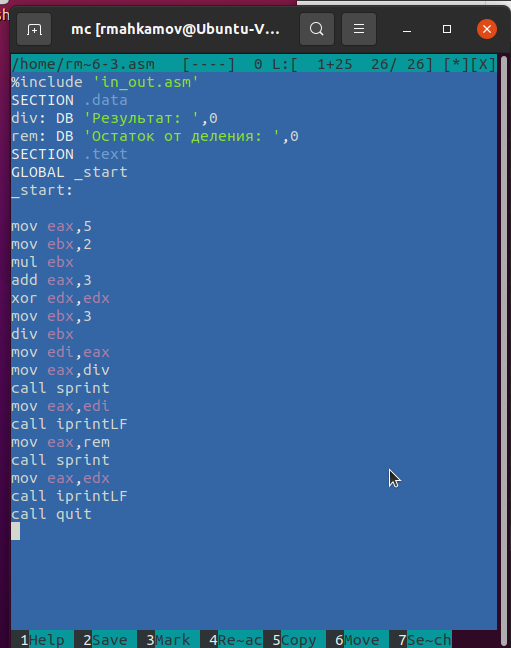


Рис. 10: Программа lab6-3.asm

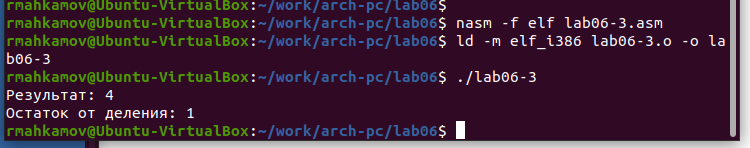


Рис. 11: Запуск программы lab6-3.asm

Изменил текст программы для вычисления выражения $ f(x) = (4 \* 6 + 2)/5 $. Создал исполняемый файл и проверил его работу. (изображение 12) (изображение 13)

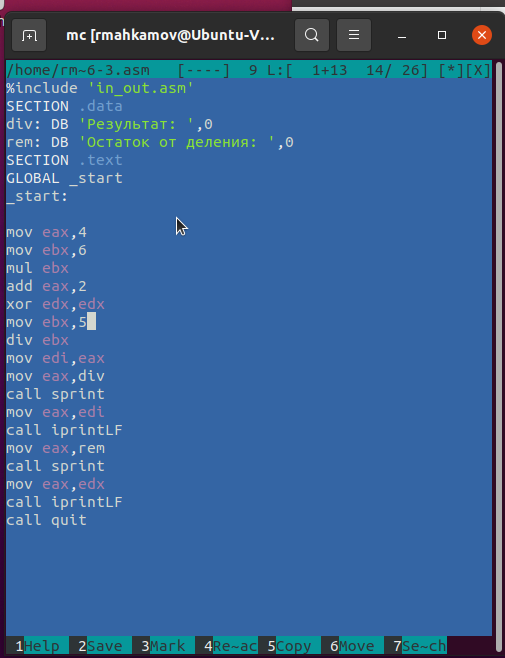


Рис. 12: Программа lab6-3.asm с другим выражением

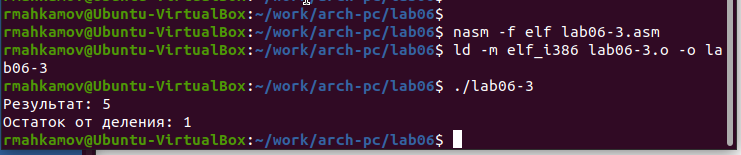


Рис. 13: Запуск программы lab6-3.asm с другим выражением

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета. (изображение 14) (изображение 15)

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.

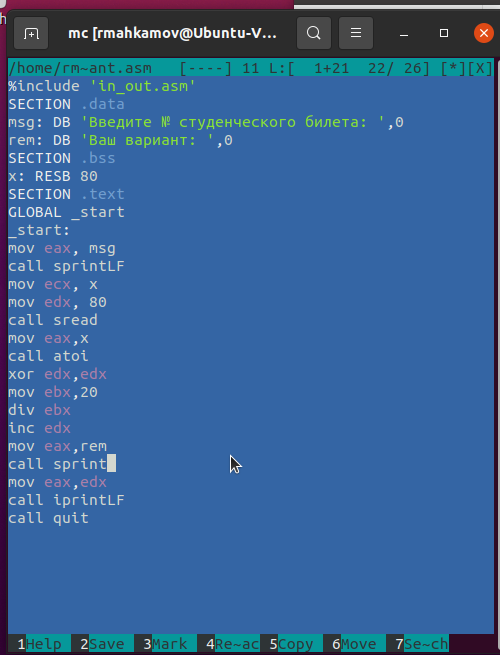


Рис. 14: Программа variant.asm

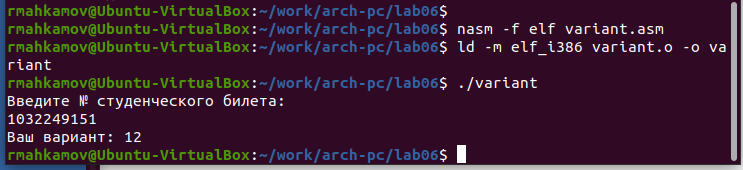


Рис. 15: Запуск программы variant.asm

## 3.3 Ответы на вопросы

1. Какие строки листинга отвечают за вывод на экран сообщения ‘Ваш вариант:’?
   * Инструкция mov eax, rem перекладывает значение переменной с фразой ‘Ваш вариант:’ в регистр eax.
   * Инструкция call sprint вызывает подпрограмму для вывода строки.
2. Для чего используются следующие инструкции?
   * 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.
3. Для чего используется инструкция “call atoi”?
   * Инструкция “call atoi” используется для преобразования введенных символов в числовой формат.
4. Какие строки листинга отвечают за вычисления варианта?
   * Инструкция xor edx, edx обнуляет регистр edx.
   * Инструкция mov ebx, 20 записывает значение 20 в регистр ebx.
   * Инструкция div ebx выполняет деление номера студенческого билета на 20.
   * Инструкция inc edx увеличивает значение регистра edx на 1.

* Здесь происходит деление номера студенческого билета на 20. В регистре edx хранится остаток, к нему прибавляется 1.

1. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?
   * Остаток от деления записывается в регистр edx.
2. Для чего используется инструкция “inc edx”?
   * Инструкция “inc edx” используется для увеличения значения в регистре edx на 1, согласно формуле вычисления варианта.
3. Какие строки листинга отвечают за вывод на экран результата вычислений?
   * Инструкция mov eax, edx перекладывает результат вычислений в регистр eax.
   * Инструкция call iprintLF вызывает подпрограмму для вывода значения на экран.

## 3.4 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. (изображение 16) (изображение 17)

Получили вариант 12 - $ (8x - 6)/2 $ для $ x=1, x=5 $

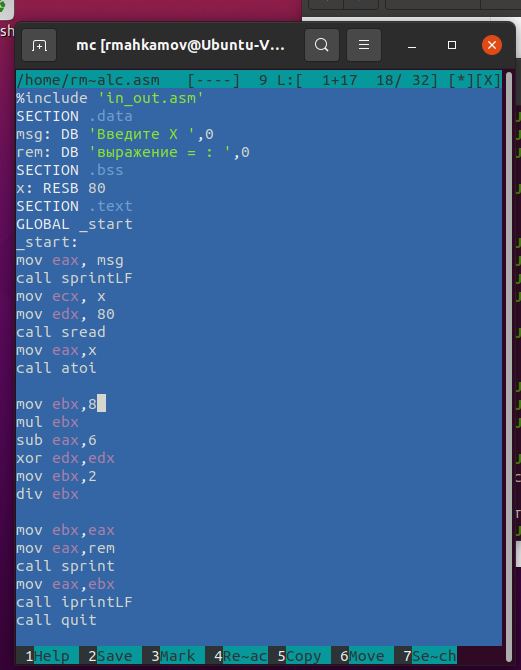


Рис. 16: Программа calc.asm

При $ x=1 $ получается 1.

При $ x=5 $ получается 17.

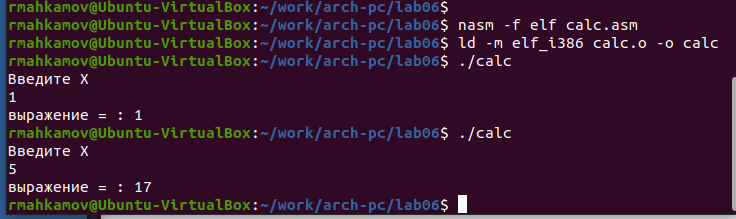


Рис. 17: Запуск программы calc.asm

Программа считает верно.

# 4 Выводы

Изучили работу с арифметическими операциями.