Отчёт по лабораторной работе 6

Арифметические операции в NASM.

Зиборова Вероника Николаевна НММбд-02-24

Содержание

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

# 2 Выполнение лабораторной работы

## 2.1 Символьные и численные данные в NASM

Создала каталог для программ лабораторной работы № 6, перешла в него и создала файл lab6-1.asm.

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

В данной программе в регистр eax записывается символ 6 (mov eax, ‘6’), в регистр ebx — символ 4 (mov ebx, ‘4’). Затем прибавляем значение регистра ebx к eax (add eax, ebx, результат сложения запишется в eax). Далее выводим результат. Так как для работы функции sprintLF в eax должен быть записан адрес, используем дополнительную переменную: записываем значение eax в buf1 (mov [buf1], eax), затем адрес buf1 в eax (mov eax, buf1) и вызываем sprintLF.

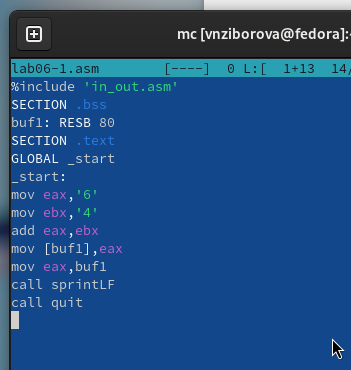


Рис. 1: Программа в файле lab6-1.asm

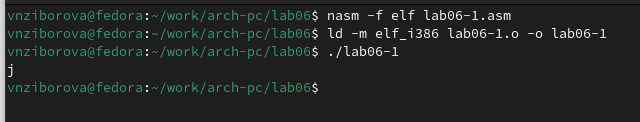


Рис. 2: Запуск программы lab6-1.asm

В данном случае при выводе значения регистра eax ожидаем увидеть число 10, но результатом будет символ j. Это происходит из-за того, что код символа 6 равен 00110110 (или 54 в десятичном представлении), а символа 4 — 00110100 (52). Команда add eax, ebx записывает в eax сумму кодов — 01101010 (106), что соответствует коду символа j.

Далее изменяю текст программы и вместо символов записываю в регистры числа.

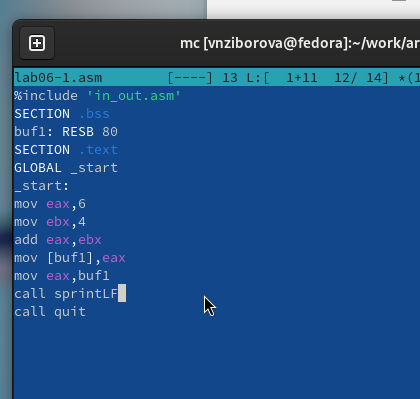


Рис. 3: Программа в файле lab6-1.asm

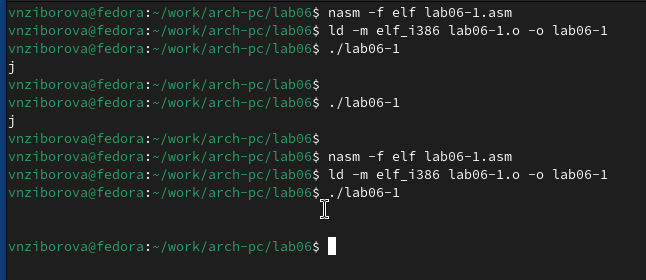


Рис. 4: Запуск программы lab6-1.asm

При исполнении программы в этом случае тоже не получаем число 10. Вместо этого выводится символ с кодом 10, который является символом конца строки и добавляет пустую строку в консоли.

Для работы с числами в файле in\_out.asm предусмотрены подпрограммы для преобразования ASCII символов в числа и обратно. Изменяю текст программы, используя эти функции.

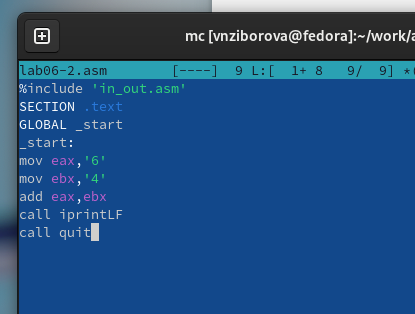


Рис. 5: Программа в файле lab6-2.asm

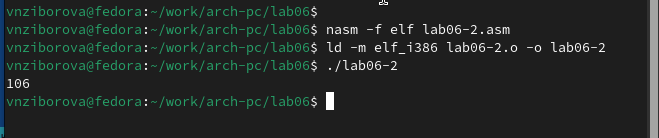


Рис. 6: Запуск программы lab6-2.asm

В результате работы программы получаем число 106. Здесь, как и в первом примере, команда add складывает коды символов 6 и 4 (54 + 52 = 106). В отличие от прошлой программы, функция iprintLF позволяет вывести число, а не символ.

Аналогично предыдущему примеру заменяю символы на числа.

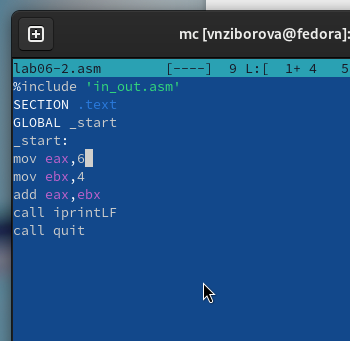


Рис. 7: Программа в файле lab6-2.asm

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

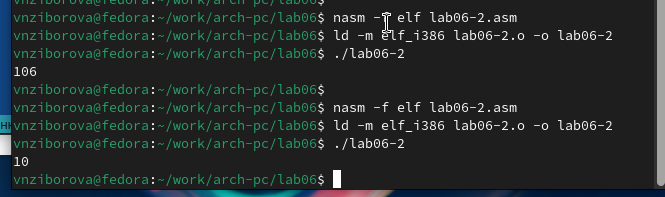


Рис. 8: Запуск программы lab6-2.asm

Заменяю функцию iprintLF на iprint, создаю исполняемый файл и запускаю его. Вывод отличается тем, что нет переноса строки.

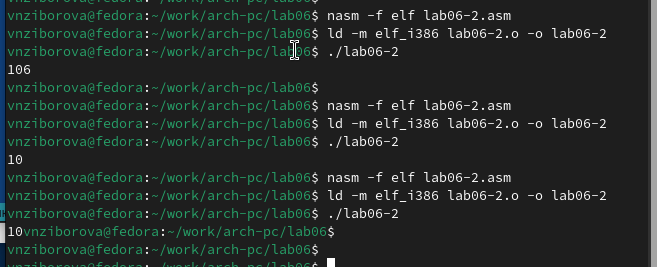


Рис. 9: Запуск программы lab6-2.asm

## 2.2 Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических операций в NASM рассмотрим программу для вычисления выражения

.

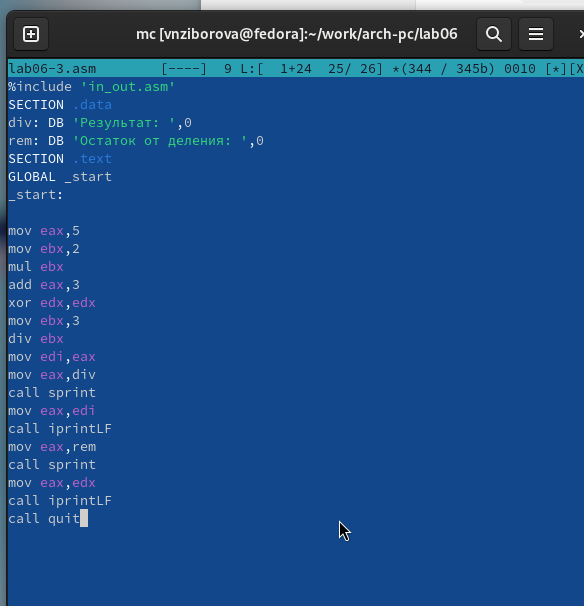


Рис. 10: Программа в файле lab6-3.asm

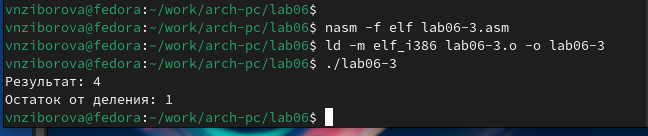


Рис. 11: Запуск программы lab6-3.asm

Изменяю текст программы для вычисления выражения

. Создаю исполняемый файл и проверяю его работу.

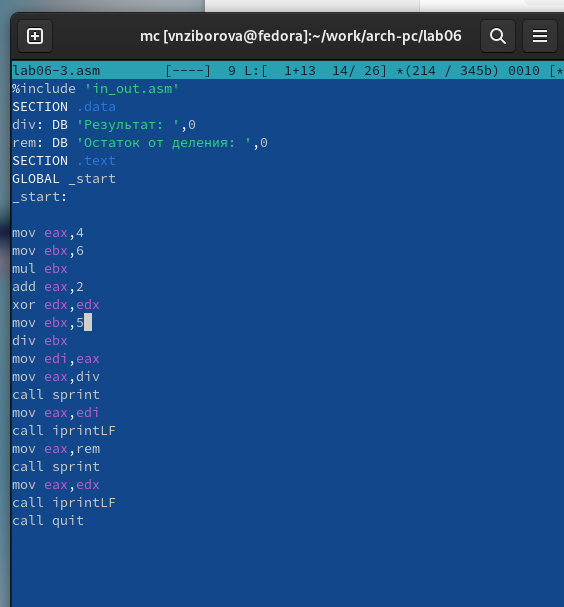


Рис. 12: Программа в файле lab6-3.asm

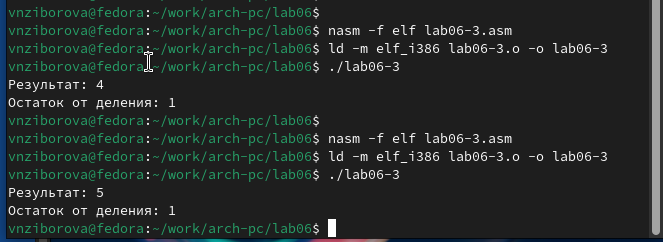


Рис. 13: Запуск программы lab6-3.asm

В качестве другого примера рассмотрим программу для вычисления варианта задания по номеру студенческого билета.

Число, над которым проводятся арифметические операции, вводится с клавиатуры. Ввод с клавиатуры осуществляется в символьном виде, и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого используется функция atoi из файла in\_out.asm.

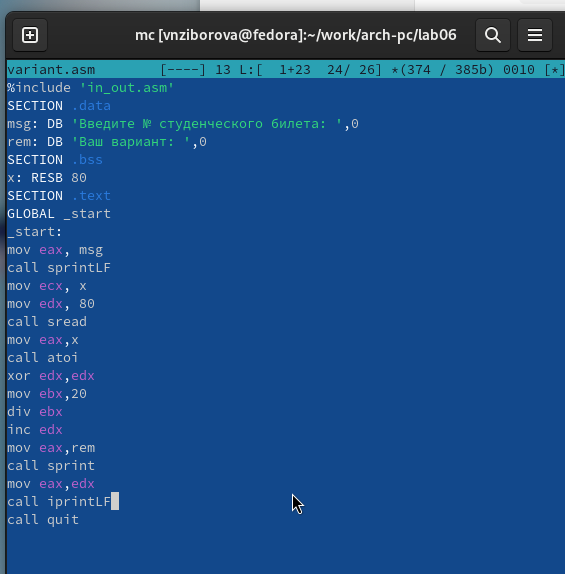


Рис. 14: Программа в файле variant.asm

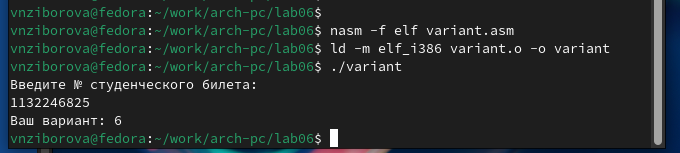


Рис. 15: Запуск программы variant.asm

### 2.2.1 Ответы на вопросы

1. Какие строки листинга отвечают за вывод сообщения ‘Ваш вариант’?
   * mov eax, rem – перекладывает в регистр значение переменной с фразой ‘Ваш вариант’.
   * call sprint – вызов подпрограммы вывода строки.
2. Для чего используются инструкции?

* mov ecx, x  
  mov edx, 80  
  call sread
  + Считывает значение студбилета в переменную Х из консоли

1. Для чего используется инструкция “call atoi”?
   * Эта подпрограмма переводит введенные символы в числовой формат.
2. Какие строки листинга отвечают за вычисления варианта?

* xor edx,edx  
  mov ebx,20  
  div ebx  
  inc edx
  + Здесь происходит деление номера студ 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”?
   * регистр edx
2. Для чего используется инструкция “inc edx”?
   * по формуле вычисления варианта нужно прибавить единицу
3. Какие строки листинга отвечают за вывод на экран результата вычислений?
   * mov eax,edx – результат перекладывается в регистр eax
   * call iprintLF – вызов подпрограммы вывода

## 2.3 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.

Получили вариант 6 -

для

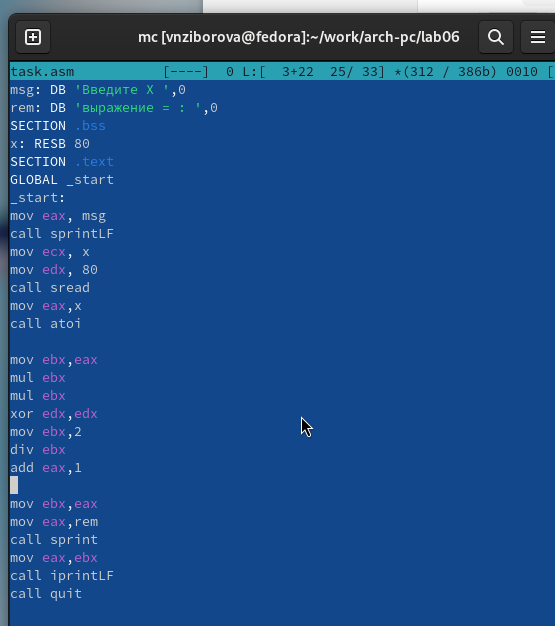


Рис. 16: Программа в файле task.asm

Если подставить 2 получается

.

Если подставить 5 получается

.

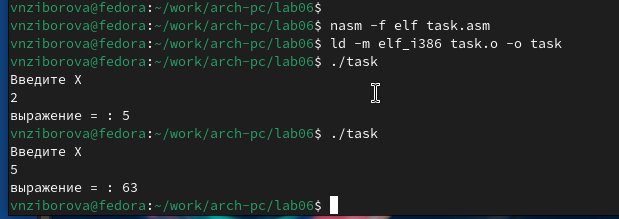


Рис. 17: Запуск программы task.asm

Программа считает верно.

# 3 Выводы

Изучили работу с арифметическими операциями.

# 4 Вопросы для самопроверки

1. **Синтаксис команды сложения чисел**: add <операнд1>, <операнд2>
2. **Команда для умножения без знака**: mul
3. **Синтаксис команды деления чисел без знака**: div <делитель>
4. **Результат при умножении двухбайтовых операндов** помещается:
   * В младшее слово регистра ax для произведения, если результат помещается в 16 бит.
   * В пару регистров dx:ax, если результат 32-битный (16-битные операнды).
5. **Арифметические команды с целочисленными операндами**:
   * add — сложение операндов.
   * sub — вычитание.
   * mul — умножение без знака.
   * imul — умножение со знаком.
   * div — деление без знака.
   * idiv — деление со знаком.
   * inc — увеличение на 1.
   * dec — уменьшение на 1.
   * neg — изменение знака операнда (отрицание).
6. **Делимое при целочисленном делении** находится в регистре ax (для 8-битных операндов) или в dx:ax (для 16-битных операндов).
7. **Неполное частное** при делении помещается в регистр ax, **остаток** — в dx.