Целые числа (в математике обозначаются как $\mathbb{Z}$, от немецкого Zahlen, числа) это:

- 0 - при сложении с ним любое целое число равно само себе
- 1 - прибавляя или отнимая 1 к/от 0 нужное число раз можно получить любое целое число

Всё, что можно получить из 0 и 1 с помощью следующих операций, тоже целые числа:

- операция сложения
- обратная ей операция вычитания
- операция умножения
- не обратная ей операция целочисленного деления, которую часто называют “делением по модулю”
- операция возведения в целую степень

Системы счисления.

Существуют разные представления чисел, а значит и разные формы их записи (системы счисления). Все используемые в информатике системы счисления позиционные, т.е. в них значение каждой цифры зависит от ее положения в числе. Например, 123 означает одна сотня два десятка и три единицы.

Если говорить именно о числах, все современные системы однородны, т.е. набор цифр, которые можно использовать в каждой позиции одинаков. Но вот, скажем, для обозначения времени используется неоднородная система счисления - часов 24, минут 60, секунд 60, миллисекунд - 1000. Из недавнего, вплоть до 1971 года британский фунт стерлингов состоял из 20 шиллингов, в каждом из которых было 12 пенсов. Впрочем, чего можно ждать от людей, которые калибр ствола орудия обозначают массой снаряда.

Примером непозиционной системы счисления является римская запись чисел, оптимизированная под сложение и вычитание небольших чисел. Например, римское 333 выглядит как CCCXXXIII, а 444 как CDXLIV, чтобы умножить их, необходимо обладать недюжинными талантами … или уметь переводить римские числа в арабские (на самом деле индийские) и обратно, да умножать столбиком.

CCCXXXIII * CDXLIV = $\overline{\text{CXLVIIDCCCLII}}$

Но вернёмся к однородным позиционным системам счисления.
Все эти они являются представлением числа в виде полинома

$$Z = A_1x^1 + A_2x^2 + \ldots + A_nx^n$$

x - фиксируется и называется основанием системы.
A_i- записываются в виде цифры в позицию с номером **i**, в привычном нам обратном порядке.

Нетрудно увидеть, что умножение числа на основание системы равносильно сдвигу на единицу счисления.

$$Z\,x = A_1x^2 + A_2x^3 + \ldots + A_nx^{n+1}$$

Так 123 * 10 превращается в 1230, а двоичное представление числа при его умножении на 2 сдвигается на разряд в старшую сторону. Разумеется, это работает даже если

проделать несколько раз. И в обратную сторону, т.е. не умножить, а разделить.

Наиболее используемыми основаниями являются
- 10, просто потому, что у нас 10 пальцев
- 2, компьютерное представление
- 8, в программировании, устаревшая
- 16, в программировании

Преобразование между системами счисления.

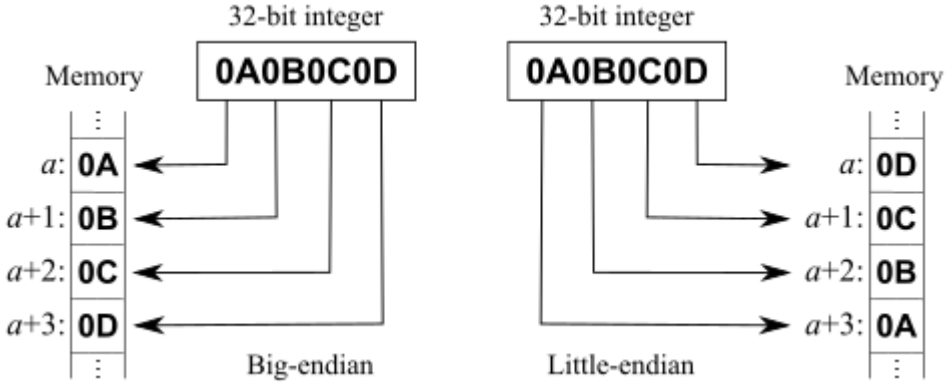
Это совсем не сложно. Для того, чтобы получить из десятичного числа восьмеричное (или шестнадцатеричное):
- берем двоичное представление десятичного числа, например
  1234567_{10} => 100101101011010000111_2
  это несложно:
  - проверяем число на четность, если нечетное, записываем в младший разряд 1, иначе 0
  - делим число на 2
  - опять проверяем результат на четность и записываем во второй разряд
  - делим результат предыдущего деления на 2
  - … и так до тех пор, пока не останется 0
- разделяем двоичное представление на группы по три разряда
  100 101 101 011 010 000 111_2
- переводим каждую группу в восьмеричную цифру
  000 => 0
  001 => 1
  010 => 2
  011 => 3
  100 => 4
  101 => 5
  110 => 6
  111 => 7
  правый разряд группы соответствует 1, средний 2, левый 4
- получаем восьмеричное представление числа
  4553207_8

В случае шестнадцатеричных чисел делить пришлось бы на группы по 4 разряда, самый левый разряд группы соответствует 8, а в качестве цифр используются латинские буквы A=10, B=11, C=12, D=13, E=14, F=15.
Таким образом 1234567_{10} => 4553207_8 => $12D687_{16}$

Представление целых чисел.

В современных компьютерах внутреннее представление двоичное, а размер регистра (компьютерное слово) является степенью двойки - 8/16/32/64. Расположение в памяти может быть различным, 32-разрядное слово, например, это 4 байта, которые могут теоретически быть расположены в памяти 24 разными способами (при условии что

биты внутри байта всегда и во всех архитектурах записываются в одном порядке, что соответствует действительности). Фактически используется два - так называемые big-endian и little endian (названия отсылают нас к Гулливеру Свифта и его остроконечникам и тупоконечникам). Некоторые архитектуры являются bi-endian (и сюда дотянулись :), т.е. могут работать с обоими представлениями.



Фиг.4.1.1 расположение целочисленных данных в памяти, отсюда

Надо сказать, двоичное представление чисел царило не всегда. На заре информатики двоичная система вполне себе конкурировала с троичной. При некотором дискомфорте для программистов, троичная система счисления позволяла использовать более простую схемотехнику. В самом деле, если в TTL логике 0V соответствует логическому нулю, а 5V - логической единице, то напряжение -5V тоже можно использовать, например, как -1 (симметричный троичный код). Или закодировать (0,1,2). При арифметических операциях происходит меньше переносов, это слегка ускоряет работу ...

Было несколько попыток создания троичных компьютеров, в частности, линейка компьютеров Сетунь использовала симметричный троичный код.
Так или иначе, должен был остаться только один способ представления данных и им стало двоичное представление. Может потому, что со временем простота схемотехники стала менее значимой, а важность удобства программирования наоборот усилилась.

8-разрядный байт и слово размером в степень двойки так же не всегда были стандартом де-факто. Так, в советской ЭВМ М-20 были 45 разрядные слова (работы со строками и байтовой адресации там не было), в БЭСМ-6 использовались 6-разрядные байты и 48-разрядные слова, в машине Минск-32 были 7-разрядные байты. В не-советских компьютерах также существовал разнобой. PDP-8: слово 12 разрядов, PDP-9: слово 18 разрядов, PDP-10: слово 36 разрядов. Только в PDP-11 появился байт в 8 разрядов и слово в 16. Этот компьютер был весьма успешен, на его архитектуре построена советская серия СМ-ЭВМ. 32-разрядное слово и 8-разрядный байт появились в серии IBM-360 (советская серия ЕС-ЭВМ). И эта архитектура (и её развитие) также оказались весьма успешны.

Пожалуй, стоит еще упомянуть такое экзотическое ныне представление чисел, как BCD. В наиболее популярном BCD представлении каждые 4 двоичных разряда представляют одну десятичную цифру. Это родовая травма, рудимент тех времён,

когда каждый уважающий себя программист должен был уметь как минимум читать автокод. А BCD представление чисел гораздо приятнее для разбора.
Между прочим, BCD представление присутствует и в архитектуре X86, что, конечно, недоразумение.