Содержание:

1. Системы счисления (хранение отрицательных чисел)
2. Числа с плавающей точкой
3. Кодировки

**Системы счисления.**

– методы записи чисел с помощью письменных знаков;

Двоичная кодировка – 0b000

Восьмеричная кодировка – число начинается с 0 (076)

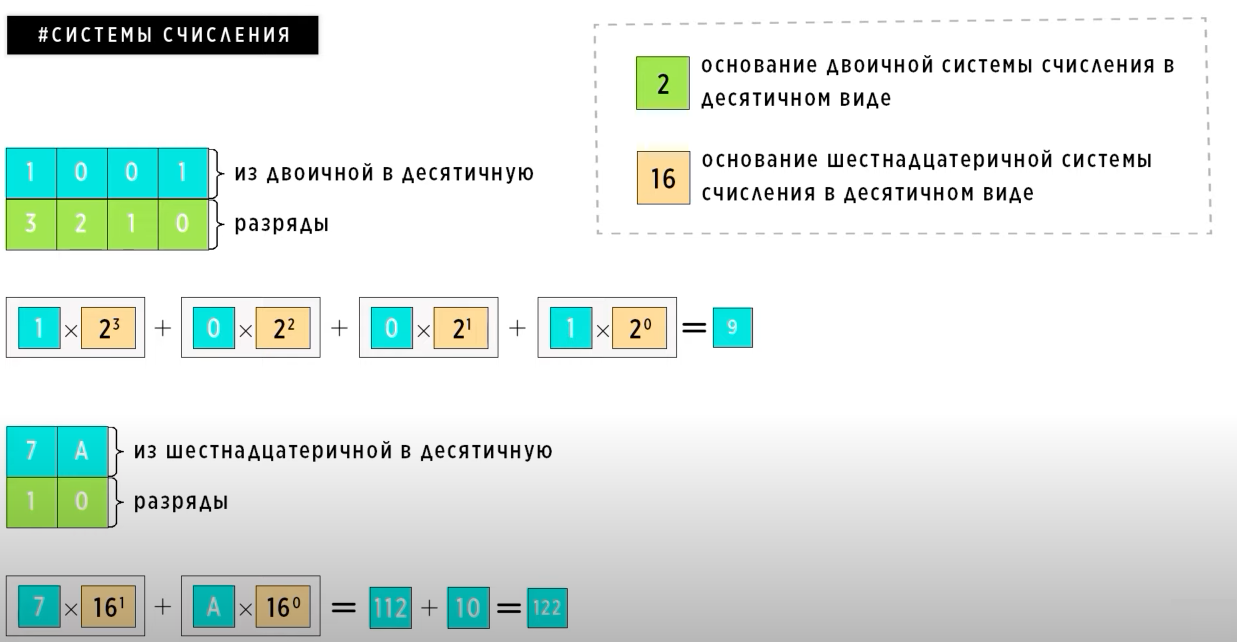
Шестнадцатеричная кодировка – 0x000

Двоичная система счисления – память компьютера состоит из битов (минимальная единица хранения инф-ии), который имеет всего 2 состояния – ток есть/тока нет, над ними надстроилась абстракция – ток есть 1, тока нет 0.

Шестнадцатеричная система - существует для упрощения чтения двоичного кода за счет того, что байт представляется в виде 2х символов в шестнадцатеричной системе (4 бита – 16 допустимых значений, что соответствует шестнадцатеричной системе) - например, байт со значением   
1010 0111 в двоичной системе соответствует А7 в шестнадцатеричной (А – число 10 в десятичной, а 7 - 7)

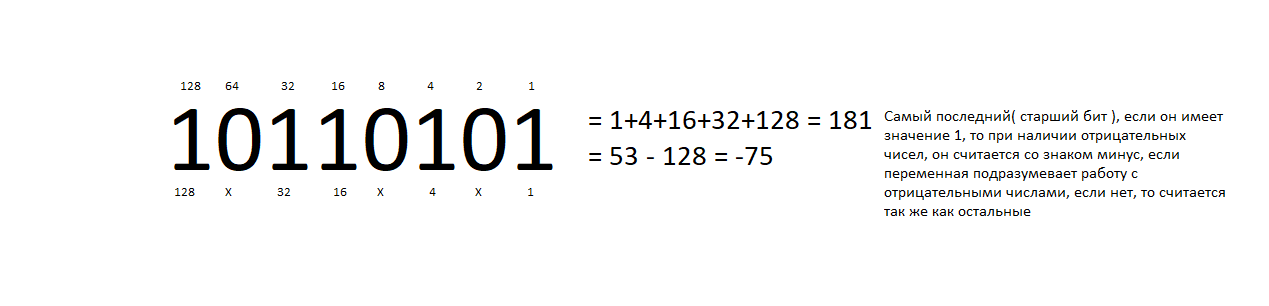
Переводы между системами:

1)Перевод через умножение

****

2)Деление с остатком – для перевода из десятичной в нужную систему (смотри рисунок)



Так же перевести число в десятичный вид из двоичного можно с помощью суммы весов каждого бита, которому соответствует 1;

*Хранение отрицательных чисел*:

Прямой код – для обозначения знака числа используется старший бит (крайний левый бит), 0 – положительное число, 1 – отрицательное. Такой код в чистом виде не используется, так как требует более сложных алгоритмов для арифметических операций (в таком варианте нельзя представить N1-N2 = N1 + (-N2)), также появляется проблема двух значащих нолей 1111 и 0000.

Обратный код – инвертированный прямой кол (все единицы меняются на 0 и наоборот) – решает проблему арифметических операций, но не для всех чисел.

Дополнительный код – прибавление к обратному коду единицы – решает проблему 2х нолей и полностью решает проблему арифметических операций.

\*Комп не умеет вычитать, он только суммирует, а операция вычитания представляется как N1-N2 = N1 + (-N2) \*

Формула макс кол-ва значений – 2^n, где n – размер регистра

Формула максимального положительного значения, хранимого в регистре – (2^n – 1) - 1, n – размер регистра;

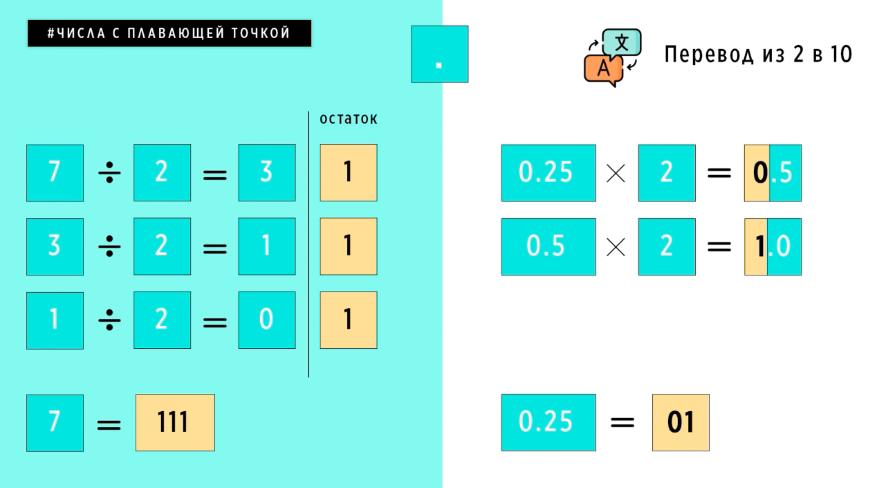
**Числа с плавающей точкой**

Стандарт хранения таких чисел IEEE-754 (Institute of Electrical and Electronics Engineers);

Стандарт хранения таких чисел не покрывает проблему математики – наличия неделимых чисел (числа с периодами, уходящие в бесконечность), поэтому в память компьютера записывается только часть этого числа, следовательно, образуется погрешность;

Есть несколько форматов точности – одинарный (float - 32 бита), двойной (double - 64бита), четырехкратный (long double - 128 байт), улучшенный двойной (long double - 80 байт);

Перевод из десятичной записи в двоичную – целая часть делением, дробная умножением и записью целой части, деление производится до тех пор, пока не получится 0 в дробной части;



Получаем 111.01 в двоичном виде;

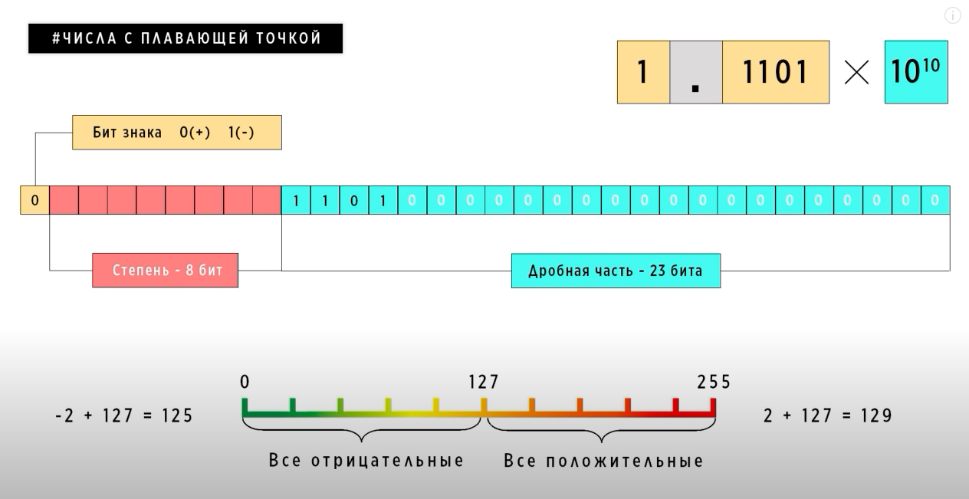
Формат записи подобных чисел в памяти соответствует экспоненциальной:

(-1)S \* 1.M \* 10E , где S – знак числа, M – Мантисса (дробная часть), E – экспонента (степень основания – кол-во цифр смещения точки от-но начального положения).

Например, 111.01 = 1.1101E2, а число 0.0011 = 1.1E-3 смысл в смещении точки, чтобы в целой части осталась единица.

Разобрав число 111.01 мы получим 1.1101E2 – степень тоже переводим в двоичный вид и получаем 1.1101E10;

Хранение:



Старший бит – знаковый бит основания, следующие 8 бит – хранят степень основания (экспонента), при этом запись степени хранится как сумма степени и среднего значения диапазона (для удобства сравнения – если диапазон задан значениями [1, 254], то его середина 127, если значение суммы меньше 127, то степень отрицательная. Значение 255 отводится под бесконечность, минус бесконечность, NaN; Минус бескон и бескон отличаются только старшим битом, оба имеют в экспоненте все единицы, а в мантиссе все 0; NaN – все тоже самое, но в мантиссе может иметь любые биты;), остальные биты отводятся под хранение дробной части (мантиссы) при этом начинается заполнение с левого бита мантиссы (см. рисунок)

Так как запись числа имеет экспоненциальный вид, то хранение целой части 1 и основания 10 не имеет смысла, так как они всегда подразумеваются в подобной записи.

***Кодировки*** – соответствие между символами (глифами), которые понимает человек с их числовыми значениями в памяти компьютера.

ASCII – первый стандарт кодирования, закодирован 1 байтом, в котором используется только 7 бит – позволял закодировать 128 символов, первые 31 значений управляющие – используются для выполнения действий, оставшиеся отображаются на экране. 8й бит (128 значений) использовался под кодовые страницы – некое расширение, куда можно было внести собственные значения (использовался для добавления собственного алфавита для каждой страны).

UNICODE – стандарт кодирования, который представлялся минимум 2мя байтами, сейчас этот стандарт содержит более 1\_100\_000 символов (код поинтов). При этом первые 128 символов полностью совпадали с ASCII, что обеспечивает обратную совместимость с ним.

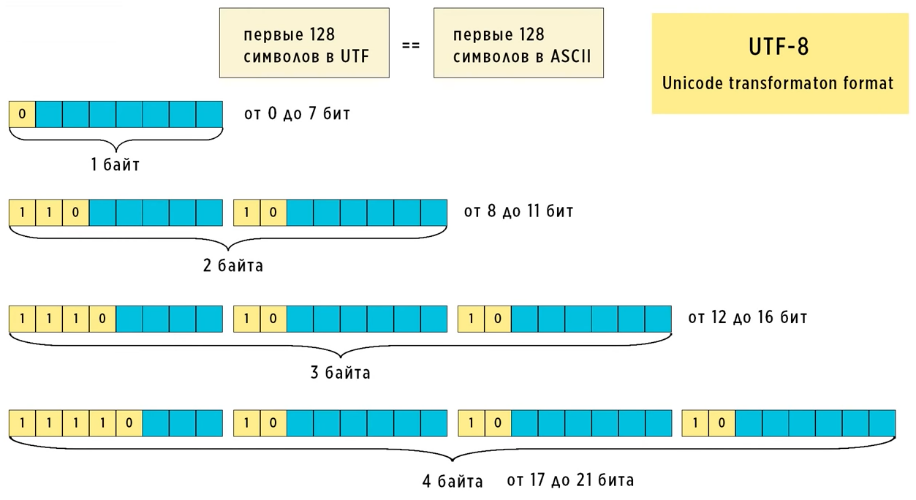
Код поинт – некое абстрактное число, которое соответствует каждому символу в этом стандарте кодировки.

Проблема Unicode:

Различные архитектуры имеют отличающийся порядок организации байт в многобайтных данных (big-endian и little-endian), что приводит к неоднозначности трактовки Unicode символов различными системами, для решения этой проблемы используются специальные флаги – BOM-байты, которые явно указывают на порядок байт – используются в кодировках с фиксированным значением (UTF-16, UTF-32).

**Кодировки** – правила, описывающие хранение Unicode символов в памяти.

UTF-8 (Unicode transformation format) – кодировка переменной длинны (от 1 до 4 байт), что дает более компактное хранение символов по сравнению с кодировками фиксированной длины, при этом жертвует скоростью чтения символов из-за необходимости проверки кол-ва байт, которые занимает символ. Не использует BOM-байты, для обозначения длины и последовательности чтения байт используются маски. (см рисунок)



UTF-16 – использует для кодирования 2 или 4 байта, имеет обратную совместимость с UCS-2 (2-байта идентичны с ней). Для хранения значений, превышающих 2 байта, используется суррогатная пара (4 байта). Для определения необходимости задействовать вторую суррогатную пару в первой имеется диапазон неиспользуемых значений, если первая суррогатная пара попадает в этот диапазон, значит нужно читать вторую. [Подробнее](https://ru.wikipedia.org/wiki/UTF-16). Также использует BOM-байты для определения порядка хранения символов в памяти. Использование суррогатной пары также доставляет некоторые сложности при чтении символов такой кодировки.

UTF-32 – не имеет динамического размера для кодирования символа, он всегда 4 байта, но при этом читается быстрее за счет того, что не нужно вычислять размер закодированного символа. Использует BOM-байты.

