Numeral Systems

In our everyday lives we use the *decimal system* to count and perform arithmetic. The decimal system is **base-ten** which means that it is based on powers of ten, or another way to look at it is that the base-ten numeral system has ten unique digits: 0 - 9.

As we count from zero, we never encounter a digit called "ten", we simply increase the number of digits and increment values from right to left:

Add 1 to 9 and the # of digits increases to 2:
9 ->becomes->10
Add 1 to 19 and the # of digits is the same, but as we "roll over" the 9 to 0, we increase the next digit to the left by 1:
19 ->becomes->20

Another way to look at it is the number of digits in a base ten system is related to the powers of ten. You've seen this with scientific notation. Where the number 1 = 1x10^0, 10 = 1x10^1, 100 = 1x10^2 and so on.

Binary Numbers

In a computer system, we use a *base-two* numeral system. We call these **binary numbers**. In our base-two system, we have two digits: 0 and 1, so we count up from zero the same as we do in base-10 but we must "roll over" every two numbers instead of every ten numbers:

```
   BASE-10              BASE-2
   0                    0
   1                    1
   2                    10 (roll over)
   3                    11
   4                    100 (roll over)
   5                    101
   6                    110 (roll over)
   7                    111
   8                    1000 (roll over)
   9                    1001
  10 (roll over)        1010 (roll over)
```

Hexidecimal Numbers

Neither binary nor decimal are convenient when describing patterns in computer system. Conversion between decimal and binary is too tedious for efficient debugging and spot-checking. Binary numbers are a natural way to decribe bit patterns but on a 32-bit system, that is 32 digits required to represent many values in the computer system. Is it convenient to write numbers like this:
11011011111111011101111100011010
MAYBE IF YOU ARE A COMPUTER!!! Imagine if you are on a 64-bit system!

Instead we use a base-16 numeral system called **hexidecimal**(*hex* for short) numbers to easily represent bit patterns in computer systems. Conversion between hexidecimal and binary is very simple and conversion between hexidecimal and decimal is much easier than converting binary to decimal. The digits of the hexidecimal numbering system are 0-9 followed by A-F:

```
        decimal         binary                  hexidecimal
        0                       0                       0
        1                       1                       1
```

2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
18	10010	12
19	10011	13
20	10100	14

If you notice 20 in decimal is 14 in hexidecimal. To clarify that we are using base-16, we write hexidecimal numbers with 0x before the digits:

> 20 in decimal is 0x14 in hexidecimal.
> 1111 in binary is is 0xF in hexidecimal.

Conversions

PRO TIP: Memorize this table and your life as a programmer will be made easier:

decimal	hex	binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Hex-to-Binary

Converting a Hexidecimal number to its binary equivalent is very straightforward because we can do it one digit at a time. The easiest approach is to reference, derive or commit to memory the 4-bit binary equivalent of each hex digit, 0- 9: (*Note: You can drop leading zeros, but not trailing zeros*)

To convert, we simply walk through the hex digtis directly substituing the four bit binary number equivalent of each hex digit.

Examples
Convert the following hex numbers to their binary equivalent:

1. 0x29FFAC04
   -> 101001111111111010110000000100
2. 0xFCFC000F
   -> 11111100111111000000000000001111

Binary-to-Hex

Binary to Hex conversions are just as straightforward. We begin by moving right-to-left and seprating the bits into groups of four, adding leading zeros if needed:

11011000100001110 -> 0001 1011 0001 0000 1110

Now, we can easily substitute the hex equivalents directly:

0001 1011 0001 0000 1110 -> 0x1B10E

Examples
Convert the following binary numbers to their hex equivalent:

1. 1011110
   -> 0x2E
2. 111110101111101000000110
   -> 0xFAFA06

Decimal-to-Hex

To convert a decimal number to its hex equivalent, we recursively divide the number *n* by 16, producing a quotient (*q*) and a remainder (*r*), such that x = q * 16 + r. We continue the recursive computation until the quotient is equal to zero. We construct the hex representation using the values of the remainder, *r*, from left-to-right (least significant digit-to-most significant digit).

Example:Convert 314156 to hexidecimal:

```
314156 / 16 = 19634 r12
19634 / 16 = 1227 r2
1227 / 16 = 76 r11
76 / 16 = 4 r12
    4 / 16 = 0 r4


4  12 11  2  12
4   C  B  2   C

  314156 = 0x4CB2C
```

Hex-to-Decimal

To convert hex to decimal, move from left-to-right multiplying the digit by the power of 16 equal to its position indexed from most-significant to least-significant digit.

Example: Convert 0x7AF to its decimal equivalent:

```
0x7 = 7  0xA = 10   0xF = 15

= (7 * 16^2) + (10 * 16^1) + (15 * 16^0)
= (7 * 256) + ( 10 * 16) + ( 15 * 1)
= (1792) + (160) + (15)
= 1967
```

Decimal-to-Binary

We convert a decimal number to a binary number in a manner similar to the decimal to hex conversion, only instead of divinding by 16 we divide by 2:

Example:Convert 56903 to binary:

```
        56903 / 2 = 28451 r1
            28451 / 2 = 14225 r1
            14225 / 2 = 7112 r1
            7112 / 2 = 3556 r0
            3556 / 2 = 1778 r0
            1778 / 2 = 889 r0
              889 /2 =  444 r1
             444 / 2 = 222 r0
             222 / 2 = 111 r0
             111 / 2 = 55 r1
               55 / 2 = 27 r1
               27 / 2 = 13 r1
               13/ 2 = 6 r1
                6 / 2 = 3 r0
                3/2 = 1 r1
                1/2 = 0 r1

 1101111001000111
```

Binary-to-Decimal

Binary to decimal is performed the same way we convert hex to decimal, only we use powers of 2 instead of powers of 16:

Example: Convert 10110011 to decimal:

```
10110011

= (1 * 2^7) + (0 * 2^6) + (1 * 2^5) + (1 * 2^4) + (0 * 2^ 3) + (0 * 2^2) + (1 * 2^1) + (1 * 2^ 0)
= ( 1 * 128) + (0 * 64) + ( 1 * 32) + ( 1 * 16) + ( 0 * 8 ) + ( 0 * 4) + ( 1 * 2) + ( 1 * 1)
= (128) + (0) + (32) + (16) + (0) + (0) + (2) + (1)
=  179
```