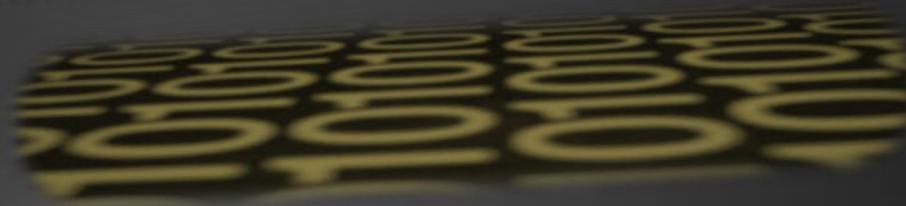


<http://csharpfundamentals.telerik.com>



# Numeral Systems

## Binary, Decimal and Hexadecimal Numbers

---

**Telerik Software Academy**  
Learning & Development Team  
<http://academy.telerik.com>



## 1. Numeral Systems

- ◆ Binary and Decimal Numbers
- ◆ Hexadecimal Numbers
- ◆ Conversion between Numeral Systems



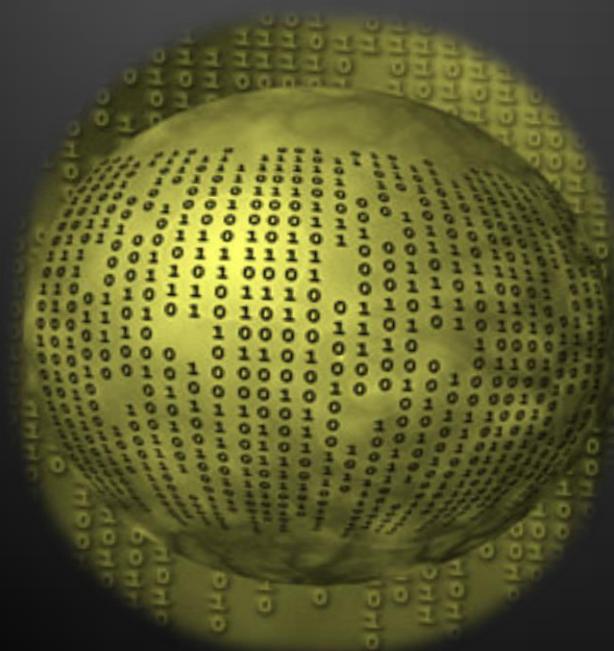
## 3. Representation of Numbers

- ◆ Positive and Negative Integer Numbers
- ◆ Floating-Point Numbers

## 4. Text Representation

# Numeral Systems

Conversion between Numeral Systems



# Decimal Numbers

- ◆ Decimal numbers (base 10)



- Represented using 10 numerals:

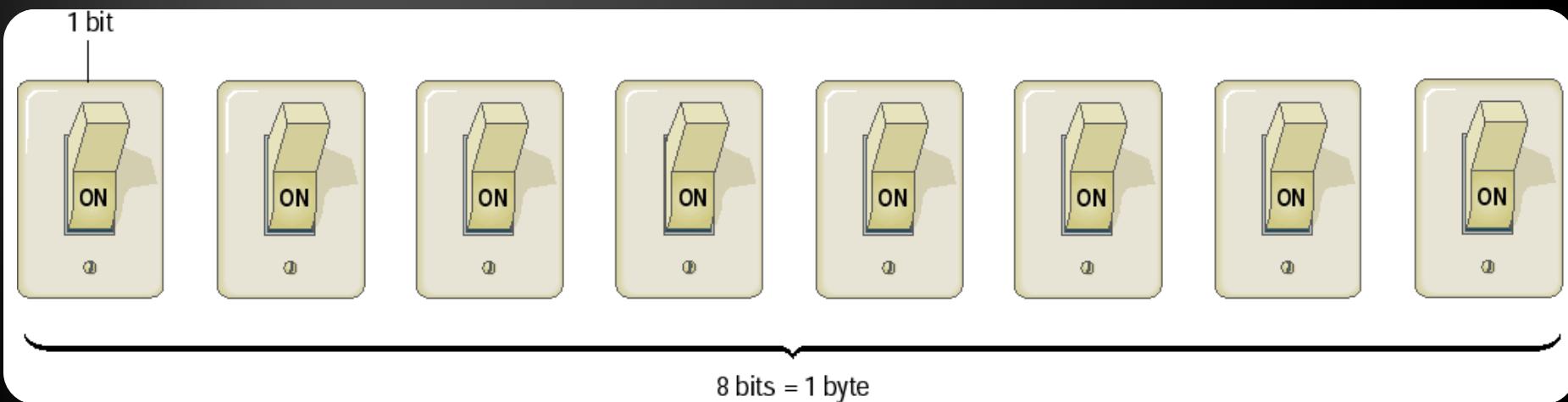
- $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$

- ◆ Each position represents a power of 10:

- $\bullet \textcolor{teal}{4} \textcolor{violet}{0} \textcolor{brown}{1} = \textcolor{teal}{4} * 10^2 + \textcolor{violet}{0} * 10^1 + \textcolor{brown}{1} * 10^0 = \textcolor{teal}{400} + \textcolor{brown}{1}$
  - $\bullet \textcolor{teal}{1} \textcolor{violet}{3} \textcolor{brown}{0} = \textcolor{teal}{1} * 10^2 + \textcolor{violet}{3} * 10^1 + \textcolor{brown}{0} * 10^0 = \textcolor{teal}{100} + \textcolor{violet}{30}$
  - $\bullet \textcolor{teal}{9} \textcolor{violet}{7} \textcolor{brown}{8} \textcolor{teal}{6} = \textcolor{orange}{9} * 10^3 + \textcolor{teal}{7} * 10^2 + \textcolor{violet}{8} * 10^1 + \textcolor{brown}{6} * 10^0 =$   
 $= \textcolor{orange}{9} * 1000 + \textcolor{teal}{7} * 100 + \textcolor{violet}{8} * 10 + \textcolor{brown}{6} * 1$

# Binary Numeral System

- ◆ Binary numbers are represented by sequence of bits (smallest unit of information – 0 or 1)
  - ◆ Bits are easy to represent in electronics



1    1    1    1    1    1    1    1

# Binary Numbers



- ◆ Binary numbers (base 2)
  - ◆ Represented by 2 numerals: 0 and 1
  - ◆ Each position represents a power of 2:
    - ◆  $101_b = 1*2^2 + 0*2^1 + 1*2^0 = 100_b + 1_b = 4 + 1 = 5$
    - ◆  $110_b = 1*2^2 + 1*2^1 + 0*2^0 = 100_b + 10_b = 4 + 2 = 6$
    - ◆  $110101_b = 1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 32 + 16 + 4 + 1 = 53$

# Binary to Decimal Conversion

- ◆ Multiply each numeral by its exponent:

- $1001_b = 1*2^3 + 1*2^0 = 1*8 + 1*1 =$   
 $= 9$

- $0111_b = 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0 =$   
 $= 100_b + 10_b + 1_b = 4 + 2 + 1 =$   
 $= 7$

- $110110_b = 1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 =$   
 $= 100000_b + 10000_b + 100_b + 10_b =$   
 $= 32 + 16 + 4 + 2 =$   
 $= 54$

# Decimal to Binary Conversion

- ◆ Divide by 2 and append the reminders in reversed order:

$$500/2 = 250 \text{ (0)}$$

$$250/2 = 125 \text{ (0)}$$

$$125/2 = 62 \text{ (1)}$$

$$62/2 = 31 \text{ (0)} \longrightarrow 500_d = 111110100_b$$

$$31/2 = 15 \text{ (1)}$$

$$15/2 = 7 \text{ (1)}$$

$$7/2 = 3 \text{ (1)}$$

$$3/2 = 1 \text{ (1)}$$

$$1/2 = 0 \text{ (1)}$$



# Hexadecimal Numbers

- ◆ Hexadecimal numbers (base 16)

- ◆ Represented using 16 numerals:  
**0, 1, 2, ... 9, A, B, C, D, E and F**

- ◆ Usually prefixed with **0x**

0 → 0x0	8 → 0x8
1 → 0x1	9 → 0x9
2 → 0x2	10 → 0xA
3 → 0x3	11 → 0xB
4 → 0x4	12 → 0xC
5 → 0x5	13 → 0xD
6 → 0x6	14 → 0xE
7 → 0x7	15 → 0xF



# Hexadecimal Numbers (2)

- ◆ Each position represents a power of 16:

- ◆  $9786_{\text{hex}} = 9 * 16^3 + 7 * 16^2 + 8 * 16^1 + 6 * 16^0 =$   
 $= 9 * 4096 + 7 * 256 + 8 * 16 + 6 * 1 =$   
 $= 38790$
- ◆  $0xABCDEF_{\text{hex}} = 10 * 16^5 + 11 * 16^4 + 12 * 16^3 +$   
 $13 * 16^2 + 14 * 16^1 + 15 * 16^0 =$   
 $= 11259375$



# Hexadecimal to Decimal Conversion

- Multiply each digit by its exponent

- $1F4_{\text{hex}} = 1*16^2 + 15*16^1 + 4*16^0 =$

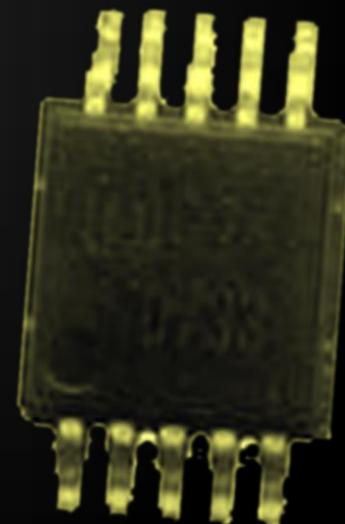
$$= 1*256 + 15*16 + 4*1 =$$

$$= 500_d$$

- $FF_{\text{hex}} = 15*16^1 + 15*16^0 =$

$$= 240 + 15 =$$

$$= 255_d$$



# Decimal to Hexadecimal Conversion

- ◆ Divide by 16 and append the reminders in reversed order

$$500/16 = 31 \text{ (4)}$$

$$31/16 = 1 \text{ (F)} \longrightarrow 500_d = 1F4_{\text{hex}}$$

$$1/16 = 0 \text{ (1)}$$



# Binary to Hexadecimal (and Back) Conversion

- The conversion from binary to hexadecimal (and back) is straightforward: each hex digit corresponds to a sequence of 4 binary digits:

**0x0 = 0000**

**0x8 = 1000**

**0x1 = 0001**

**0x9 = 1001**

**0x2 = 0010**

**0xA = 1010**

**0x3 = 0011**

**0xB = 1011**

**0x4 = 0100**

**0xC = 1100**

**0x5 = 0101**

**0xD = 1101**

**0x6 = 0110**

**0xE = 1110**

**0x7 = 0111**

**0xF = 1111**



# Numbers Representation

Positive and Negative Integers and  
Floating-Point Numbers

# Representation of Integers

- ◆ A short is represented by 16 bits

- ◆  $100 = 2^6 + 2^5 + 2^2 =$   
 $= 00000000 \ 01100100$

- ◆ An int is represented by 32 bits

- ◆  $65545 = 2^{16} + 2^3 + 2^0 =$   
 $= 00000000 \ 00000001 \ 00000000 \ 00001001$

- ◆ A char is represented by 16 bits

- ◆  $'0' = 48 = 2^5 + 2^4 =$   
 $= 00000000 \ 00110000$



# Positive and Negative Numbers

- ◆ A number's sign is determined by the Most Significant Bit (MSB)
  - Only in signed integers: `sbyte`, `short`, `int`, `long`
  - Leading 0 means positive number
    - Leading 1 means negative number
  - Example: (8 bit numbers)

$0XXXXXXX_b > 0$       e.g.  $00010010_b = 18$

$00000000_b = 0$

$1XXXXXXX_b < 0$       e.g.  $10010010_b = -110$

# Positive and Negative Numbers (2)

- ◆ The largest positive 8-bit sbyte number is:  
 $127 (2^7 - 1) = 01111111_b$
- ◆ The smallest negative 8-bit number is:  
 $-128 (-2^7) = 10000000_b$
- ◆ The largest positive 32-bit int number is:  
 $2\ 147\ 483\ 647 (2^{31} - 1) = 0111...11111_b$
- ◆ The smallest negative 32-bit number is:  
 $-2\ 147\ 483\ 648 (-2^{31}) = 1000...00000_b$

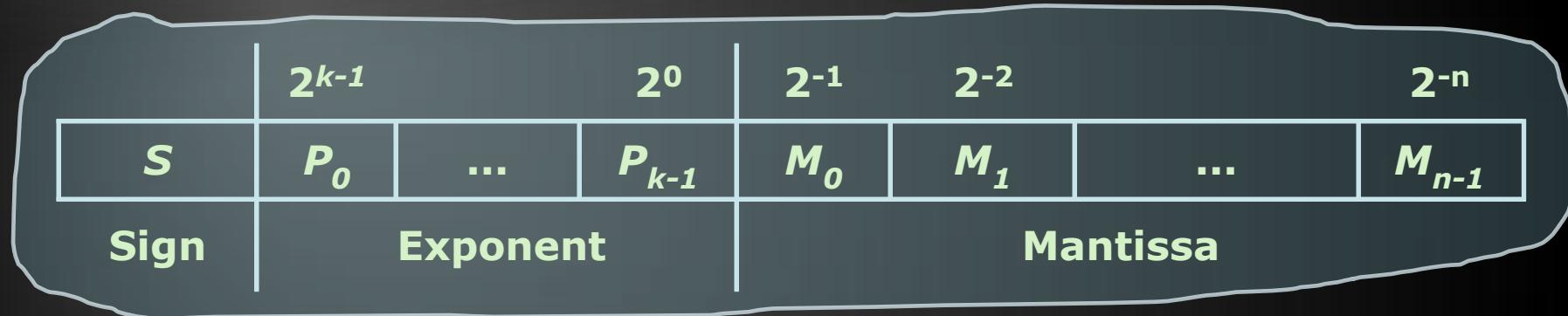
# Representation of 8-bit Numbers

- Positive 8-bit numbers have the format **0XXXXXXX**
  - Their value is the decimal of their last 7 bits (XXXXXXX)
- Negative 8-bit numbers have the format **1YYYYYYY**
  - Their value is 128 ( $2^7$ ) minus (-) the decimal of YYYYYYY
  - $10010010_b = 2^7 - 10010_b = 128 - 18 = -110$

+127	=	01111111
...		
+3	=	00000011
+2	=	00000010
+1	=	00000001
+0	=	00000000
-1	=	11111111
-2	=	11111110
-3	=	11111101
...		
-127	=	10000001
-128	=	10000000

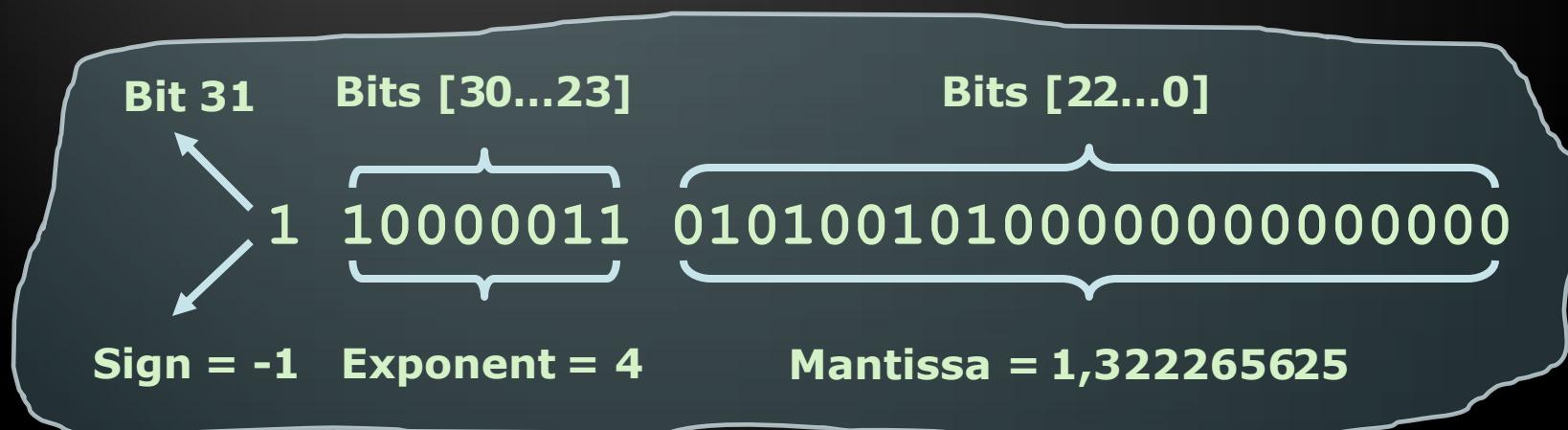
# Floating-Point Numbers

- ◆ Floating-point numbers representation (according to the IEEE 754 standard\*):



- ◆ Example:

\* See [http://en.wikipedia.org/wiki/Floating\\_point](http://en.wikipedia.org/wiki/Floating_point)





# Text Representation in Computer Systems

# How Computers Represent Text Data?

- ◆ A text encoding is a system that uses binary numbers (1 and 0) to represent characters
  - Letters, numerals, etc.
- ◆ In the ASCII encoding each character consists of 8 bits (one byte) of data
  - ASCII is used in nearly all personal computers
- ◆ In the Unicode (UTF-16) encoding each character consists of 16 bits (two bytes)
  - Can represent many alphabets

# Character Codes – ASCII Table

Excerpt  
from the  
ASCII  
table

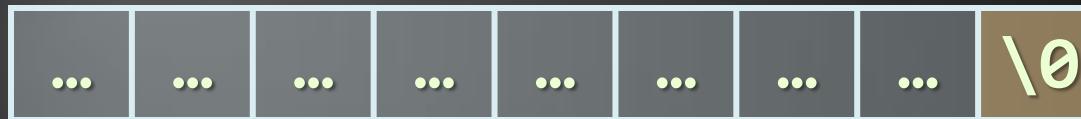
^	<^>	<^>	^
==	=	=	==
==	^	==	
---	*	-	-
000000	8888888		
00000000	8	88	
00000000		88	
00!000000		88	
!!00!000000		88	
!!00!0000000		88	
!!00!00000000		88	
00 00 000000	88		
0000 0000 0000	88		

Binary Code	Decimal Code	Character
01000001	65	A
01000010	66	B
01000011	67	C
01000100	68	D
00100011	35	#
01100000	48	0
00110001	49	1
01111110	126	~

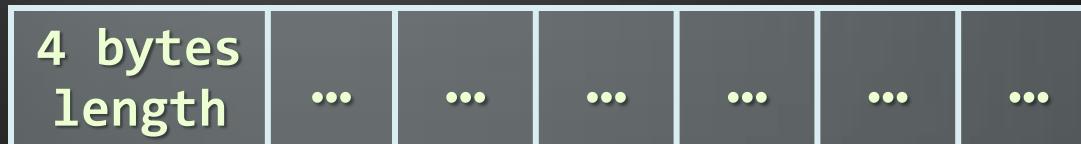
# Strings of Characters

- ◆ Strings are sequences of characters

- ◆ Null-terminated (like in C)



- ◆ Represented by array



- ◆ Characters in the strings can be:

- ◆ 8 bit (ASCII / windows-1251 / ...)
  - ◆ 16 bit (UTF-16)

# Numeral Systems



# Questions?



1. Write a program to convert decimal numbers to their binary representation.
2. Write a program to convert binary numbers to their decimal representation.
3. Write a program to convert decimal numbers to their hexadecimal representation.
4. Write a program to convert hexadecimal numbers to their decimal representation.
5. Write a program to convert hexadecimal numbers to binary numbers (directly).
6. Write a program to convert binary numbers to hexadecimal numbers (directly).

7. Write a program to convert from any numeral system of given base  $s$  to any other numeral system of base  $d$  ( $2 \leq s, d \leq 16$ ).
8. Write a program that shows the binary representation of given 16-bit signed integer number (the C# type short).
9. \* Write a program that shows the internal binary representation of given 32-bit signed floating-point number in IEEE 754 format (the C# type float).  
Example: -27,25 → sign = 1, exponent = 10000011, mantissa = 101101000000000000000000.

# Free Trainings @ Telerik Academy

- ◆ “C# Programming @ Telerik Academy

- ◆ [csharpfundamentals.telerik.com](http://csharpfundamentals.telerik.com)



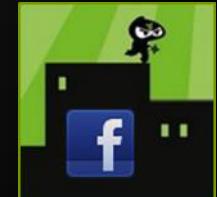
- ◆ Telerik Software Academy

- ◆ [academy.telerik.com](http://academy.telerik.com)



- ◆ Telerik Academy @ Facebook

- ◆ [facebook.com/TelerikAcademy](https://facebook.com/TelerikAcademy)



- ◆ Telerik Software Academy Forums

- ◆ [forums.academy.telerik.com](http://forums.academy.telerik.com)

