# Number Systems

CIS-2151

Peter Chapin

Vermont Technical College

# Decimal (base 10)

- Ten digits 0 → 9
- Each digit has 10x the weight of the digit to its right
- 1234… ‘1’ is the “most significant” digit; ‘4’ is the “least significant”
- $1234 = 1*10^3 + 2*10^2 + 3*10^1 + 4*10^0 = 1234$
- Shifting to the left multiples by ten: 12340
- Shifting to the right divides by ten: 123 (remainder, 4, is lost).

# Binary (base 2)

- Two digits 0 → 1
- Each digit has 2x the weight of the digit to its right
- 1010… the leftmost ‘1’ is the “most significant” binary digit (bit), also called the MSB; the rightmost ‘0’ is the “least significant” bit, also called the LSB.
- $1010 = 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = 8 + 2 = 10$
- Shifting to the left multiples by two: 10100 = 20
- Shifting to the right divides by two: 101 = 5 (with remainder 0).

# Powers of Two

| Power of Two | Binary | Decimal |
|---|---|---|
| $2^0$ | 0000 0001 | 1 |
| $2^1$ | 0000 0010 | 2 |
| $2^2$ | 0000 0100 | 4 |
| $2^3$ | 0000 1000 | 8 |
| $2^4$ | 0001 0000 | 16 |
| $2^5$ | 0010 0000 | 32 |
| $2^6$ | 0100 0000 | 64 |
| $2^7$ | 1000 0000 | 128 |

# Special Powers of Two

| Power of Two | Binary | Decimal |
|---|---|---|
| $2^8$ | 0000 0000 0000 0000 0000 0001 0000 0000 | 256 |
| $2^{10}$ | 0000 0000 0000 0000 0000 0100 0000 0000 | 1,024 (“K”) |
| $2^{16}$ | 0000 0000 0000 0001 0000 0000 0000 0000 | 65,536 (“64K”) |
| $2^{20}$ | 0000 0000 0001 0000 0000 0000 0000 0000 | $(1024)^2$ = 1,048,576 (“M”) |
| $2^{30}$ | 0100 0000 0000 0000 0000 0000 0000 0000 | $(1024)^3$ = 1,073,741,824 (“G”) |

# Bytes/Octets

- An 8-bit binary value is commonly called a “byte”
  - Network people like the word “octet.” On some (very exotic) systems “byte” refers to a different number of bits.
- Smallest octet: 0000 0000 = 0
- Largest octet: 1111 1111 = 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255
- Thus the **range of octet values is 0 .. 255** (that is, 256 distinct values)
- Memory sizes and storage sizes are quoted in bytes.
  - A machine with 16 GB of memory holds (<u>about</u> 16 billion, really $16*2^{30}$, of these eight bit values in memory). *All data manipulated by your computer eventually boils down to 8-bit bytes of information!*

# Notation

- The use of ‘K’, ‘M’, and ‘G’ for “kilo-”, “mega-” and “giga-” is ambiguous.
  - Most scientists and engineers use these prefixes to mean $10^3$, $10^6$, and $10^9$.
    - Kilometer: 1000 meters
    - Kilogram: 1000 grams, etc.
  - Computer people sometimes use them to mean $2^{10}$, $2^{20}$, and $2^{30}$ instead, which are close in value to the usual meaning, but not exactly the same.
    - Kilobyte: 1024 bytes, etc.
- ISO (the International Organization for Standardization) recommends using “KiB”, “MiB”, and “GiB” for kilobyte, megabyte, and gigabyte
  - Not widely followed, but you will see it.

# Hexadecimal, aka Hex (base 16)

- Sixteen digits 0 → 9 and A → F (lower case commonly used)
- Each digit has 16x the weight of the digit to its right
- 2A3C… ‘2’ is the “most significant” digit; ‘C’ is the “least significant”
- 2A3C = $2*16^3 + 10*16^2 + 3*16^1 + 12*16^0 = 10,812$
- Shifting to the left multiples by sixteen: 2A3C0 = 172,992
- Shifting to the right divides by sixteen: 2A3 = 675 (remainder, 12, is lost).
- Hex values often prefixed with “0x” … 0x2A3C
  - Sometimes suffixed with “H” … 2A3CH (this is much less common)

# Hex to Binary

| Hex Digit | Binary |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |

| Hex Digit | Binary |
|---|---|
| 8 | 1000 |
| 9 | 1001 |
| A | 1010 |
| B | 1011 |
| C | 1100 |
| D | 1101 |
| E | 1110 |
| F | 1111 |

**Memorize This!**

# We Love Hex

- Conversion to/from binary is trivial (did you memorize the table?)
  - → Hex
    - Group the binary value into groups of 4 bits (starting from the right). Then convert each group into its hex digit using the table (that you memorized)

```
1011,0110,0000,0101,1001,0111,1011,1111
   B    6    0    5    9    7    B    F = 0xB60597BF
```

  - → Binary
    - Convert each hex digit into it's corresponding bit pattern.

```
0x1C2533E7 =    1    C    2    5    3    3    E    7
             0001,1100,0010,0101,0011,0011,1110,0111
```

  - *Hex is just a compact way to read/write large binary values* (32 bits above)
  - **An octet is just two hex digits that range from 0x00 .. 0xFF**

# Converting Decimal to Binary?

- Converting binary to/from hex is trivial (memorize the table)
- Converting from decimal to binary is trickier
  - There are various methods.
  - I like the division method.
- Instead of using fractions, we'll speak of remainders.
  - For example, 27 / 4 = 6 with remainder 3 (not 6.75).
    - Note: 6*4 + 3 = 27. We can put the number back together again from its parts.
  - If your calculator gives you 6.75…
    - Subtract the 6 like this: 6.75 – 6 = 0.75
    - Multiply the result by the original 4 like this: 0.75 * 4 = 3 (the remainder!)

# Example Decimal to Binary

- Repeatedly divide by 2. Example: 155
  - `155 / 2 = 77` with remainder 1
    `77 / 2 = 38` with remainder 1
    `38 / 2 = 19` with remainder 0 (divided evenly)
    `19 / 2 =  9` with remainder 1
    `9 / 2 =  4` with remainder 1
    `4 / 2 =  2` with remainder 0
    `2 / 2 =  1` with remainder 0
    `1 / 2 =  0` with remainder 1 (stop here, a quotient of zero reached)
- Now read off the remainders from least to most significant
  - 1 0 0 1, 1 0 1 1

# Example Decimal to Hex

- Repeatedly divide by 16. Example: 3,962
  - `3962 / 16 = 247` with remainder 10 (hex: A)
    ` 247 / 16 =  15` with remainder 7 (hex: 7)
    `  15 / 16 =   0` with remainder 15 (hex: F)
- Now read off the reminders from least to most significant
  - 0xF7A
  - Commonly we pad the left side with zeros to fill out a particular overall number of bits (where the overall size required depends on context).
    - As 16 bits: 0x0F7A
    - As 32 bits: 0x00000F7A
    - As 64 bits: 0x0000000000000F7A

# ASCII

- *Not really a number systems topic, but…*
- **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
  - A way of associating "characters" (letters, numbers, punctuation, etc.) with numbers.
  - A 7-bit code, so only $2^7 = 128$ different characters supported.
    - Control characters (used for [serial] communications and other things)
      - Includes the carriage return, new line, tab, form feed, and various others
    - Letters (upper and lower case are distinct)
    - Numeric digits (the ASCII code for the digit '3' is not 0x03. It is 0x33)
    - Punctuation marks of various kinds
  - The 8th (most significant) bit is always zero. Sometimes used as a *parity* bit.

# Extended ASCII?

- ASCII is really only good for English text
  - Doesn’t do accented letters used by various Western European languages
  - Doesn’t do non-Latin characters
    - Various eastern languages (Chinese and it’s variants, Japanese, Korean, etc.)
    - Russian
    - Hebrew
    - Arabic
    - etc., etc., etc.
- Character sets is a *BIG* subject!
  - Today people tend toward Unicode, which is a complex character set with multiple character encodings. Outside the scope of this course.