Unit 3
Binary Representation

ANALOG VS. DIGITAL

Analog vs. Digital

- The analog world is based on continuous events. Observations can take on (real) any value.
- The digital world is based on discrete events. Observations can only take on a finite number of discrete values

Q. Which is better?
A. Depends on what you are trying to do.

- Some tasks are better handled with analog data, others with digital data.
  - Analog means continuous/real valued signals with an infinite number of possible values
  - Digital signals are discrete [i.e. 1 of n values]
Analog vs. Digital

• How much money is in my checking account?
  – Analog: Oh, some, but not too much.
  – Digital: $243.67

Analog vs. Digital

• How much do you love me?
  – Analog: I love you with all my heart!!!!
  – Digital: $3.2 \times 10^3$ MegaHearts

The Real (Analog) World

• The real world is inherently analog.
• To interface with it, our digital systems need to:
  – Convert analog signals to digital values (numbers) at the input.
  – Convert digital values to analog signals at the output.
• Analog signals can come in many forms
  – Voltage, current, light, color, magnetic fields, pressure, temperature, acceleration, orientation

Digital is About Numbers

• In a digital world, numbers are used to represent all the possible discrete events
  – Numerical values
  – Computer instructions (ADD, SUB, BLE, …)
  – Characters (‘a’, ‘b’, ‘c’, …)
  – Conditions (on, off, ready, paper jam, …)
• Numbers allow for easy manipulation
  – Add, multiply, compare, store, …
• Results are repeatable
  – Each time we add the same two number we get the same result
3.9 Digital Representation

Interpreting Binary Strings

- Given a string of 1’s and 0’s, you need to know the representation system being used, before you can understand the value of those 1’s and 0’s.

01000001 = ?

65\text{_{10}}

41\text{_{BCD}}

‘A’\text{_{ASCII}}

3.10

Binary Representation Systems

- Integer Systems
  - Unsigned
    - Unsigned (Normal) binary
  - Signed
    - Signed Magnitude
    - 2’s complement
    - Excess-N*
    - 1’s complement*

- Floating Point
  - For very large and small (fractional) numbers

- Codes
  - Text
    - ASCII / Unicode
  - Decimal Codes
    - BCD (Binary Coded Decimal) / (8421 Code)

3.11

3.12

OVERVIEW

* = Not fully covered in this class
4 Skills

- We will teach you 4 skills that you should know and be able to apply with confidence
  - Convert a number in any base (base r) to decimal (base 10)
  - Convert a decimal number (base 10) to binary
  - Use the shortcut for conversion between binary (base 2) and hexadecimal (base 16)
  - Understand the finite number of combinations that can be made with n bits (binary digits) and its implication for codes including ASCII and Unicode

Number Systems

- Number systems consist of
  1. ______________
  2. ___ coefficients [__________]
- Human System: Decimal (Base 10): 0,1,2,3,4,5,6,7,8,9
- Computer System: Binary (Base 2): 0,1
- Human systems for working with computer systems (shorthand for human to read/write binary)
  - ________________
  - ________________

Anatomy of a Decimal Number

- A number consists of a string of explicit coefficients (digits).
- Each coefficient has an implicit place value which is a ______ of the base.
- The value of a decimal number (a string of decimal coefficients) is the sum of each coefficient times it place value

\[(934)_{10} = 9\times\underline{ } + 3\times\underline{ } + 4\times\underline{ } = \underline{ }\]

\[(3.52)_{10} = 3\times\underline{ } + 5\times\underline{ } + 2\times\underline{ } = \underline{ }\]
Anatomy of a Binary Number

- Same as decimal but now the coefficients are 1 and 0 and the place values are the powers of 2

```
(1011)_2 = 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0
```

General Conversion From Base r to Decimal

- A number in base r has place values/weights that are the powers of the base
- Denote the coefficients as: \( a_i \)

\[
N_r = a_3 a_2 a_1 a_0 a_{-1} a_{-2} r^3 + a_2 r^2 + a_1 r^1 + a_0 r^0 + a_{-1} r^{-1} + a_{-2} r^{-2}
\]

Examples

(746)_8 =

(1A5)\(_{16} =

(AD2)\(_{16} =

Binary Examples

(1001.1)_2 =

(10110001)_2 =
Powers of 2

\[
\begin{align*}
2^0 &= 1 \\
2^1 &= 2 \\
2^2 &= 4 \\
2^3 &= 8 \\
2^4 &= 16 \\
2^5 &= 32 \\
2^6 &= 64 \\
2^7 &= 128 \\
2^8 &= 256 \\
2^9 &= 512 \\
2^{10} &= 1024 \\
\end{align*}
\]

Unique Combinations

- Given \( n \) digits of base \( r \), how many unique numbers can be formed? __  
  
  - What is the range? \[[________\]  

<table>
<thead>
<tr>
<th>( r )</th>
<th>( n )</th>
<th>2-digit, decimal numbers</th>
<th>3-digit, decimal numbers</th>
<th>4-bit, binary numbers</th>
<th>6-bit, binary numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>0-9</td>
<td>0-9</td>
<td>0-1</td>
<td>0-1</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>0-9</td>
<td>0-9</td>
<td>0-1</td>
<td>0-1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0-1</td>
<td>0-1</td>
<td>0-1</td>
<td>0-1</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>0-1</td>
<td>0-1</td>
<td>0-1</td>
<td>0-1</td>
</tr>
</tbody>
</table>

Main Point: Given \( n \) digits of base \( r \), ___ unique numbers can be made with the range \[[________]\]

Approximating Large Powers of 2

- Often need to find decimal approximation of a large powers of 2 like \(2^{16}, 2^{32}, \text{etc.}\) \[2^{16} = 2^8 \times 2^{10} = 2^{24}\]

- Use following approximations:
  - \(2^{10} = \) __________
  - \(2^{20} = \) __________
  - \(2^{30} = \) __________
  - \(2^{40} = \) __________

- For other powers of 2, decompose into product of \(2^{10}\) or \(2^{20}\) or \(2^{30}\) and a power of 2 that is less than \(2^{10}\)
  
  - 16-bit half word: 64K numbers
  - 32-bit word: 4G numbers
  
"Making change"

BASE 10 TO BASE 2 OR BASE 16
Decimal to Unsigned Binary

- To convert a decimal number, $x$, to binary:
  - Only coefficients of 1 or 0. So simply find place values that add up to the desired values, starting with larger place values and proceeding to smaller values and place a 1 in those place values and 0 in all others.

  $25_{10} = \begin{array}{cccc} 32 & 16 & 8 & 4 \\ \end{array}$

Decimal to Unsigned Binary

- $73_{10} = \begin{array}{cccc} 128 & 64 & 32 & 16 \\ \end{array}$
- $87_{10} = \begin{array}{cccc} 64 & 32 & 16 & 8 \\ \end{array}$
- $145_{10} = \begin{array}{cccc} 128 & 64 & 32 & 16 \\ \end{array}$
- $0.625_{10} = \begin{array}{cccc} .5 & .25 & .125 & .0625 \\ \end{array}$

Decimal to Another Base

- To convert a decimal number, $x$, to base $r$:
  - Use the place values of base $r$ (powers of $r$). Starting with largest place values, fill in coefficients that sum up to desired decimal value without going over.

  $75_{10} = \begin{array}{cccc} 256 & 16 & 1 \\ \end{array}$

Shortcuts for Converting Binary ($r=2$), Hexadecimal ($r=16$) and Octal ($r=8$)

**SHORTHAND FOR BINARY**
Binary, Octal, and Hexadecimal

- Octal (base 8 = $2^3$)
  - 1 Octal digit ( _ )$_8$ can represent: ________
  - 3 bits of binary ( _ _ _)$_2$ can represent: 000-111 = ________
- Conclusion... 
  __Octal digit = __ bits

- Hex (base 16=2$^4$)
  - 1 Hex digit ( _ )$_{16}$ can represent: 0-F (_____
  - 4 bits of binary ( _ _ _ _)$_2$ can represent: 0000-1111 = ________
- Conclusion... 
  __ Hex digit = ___ bits

Binary to Octal or Hex

- Make groups of 3 bits starting from radix point and working outward
- Add 0’s where necessary
- Convert each group of 3 to an octal digit

Octal or Hex to Binary

- Expand each octal digit to a group of 3 bits
  - 317.2$_8$
- Expand each hex digit to a group of 4 bits
  - D93.8$_{16}$

Hexadecimal Representation

- Since values in modern computers are many bits, we use hexadecimal as a shorthand notation (4 bits = 1 hex digit)
  - 11010010 = D2 hex or 0xD2 if you write it in C/C++
  - 0111011011001011 = 76CB hex or 0x76CB if you write it in C/C++
Binary Codes

- Using binary we can represent any kind of information by coming up with a code
- Using $n$ bits we can represent $2^n$ distinct items

Colors of the rainbow:
- Red = 000
- Orange = 001
- Yellow = 010
- Green = 100
- Blue = 101
- Purple = 111
- Red = 000
- Orange = 001
- Yellow = 010
- Green = 100
- Blue = 101
- Purple = 111

Letters:
- 'A' = 00000
- 'B' = 00001
- 'C' = 00010
- 'Z' = 11001

BCD (If Time Permits)

- Rather than convert a decimal number to binary which may lose some precision (i.e. $0.1_{10} = \text{infinite binary fraction}$), BCD represents each decimal digit as a separate group of bits (exact decimal precision)
  - Each digit is represented as a _________ number (using place values 8,4,2,1 for each dec. digit)
  - Often used in financial and other applications where decimal precision is needed

$$439_{10}$$

BCD Representation:

Unsigned Binary Rep.: $110110111_2$

Important: Some processors have specific instructions to operate on #'s represented in BCD
3.37 ASCII Code
- Used for representing text characters
- Originally 7-bits but usually stored as 8-bits = 1-byte in a computer
- Example:
  - "Hello\n";
  - Each character is converted to ASCII equivalent
    - ‘H’ = 0x48, ‘e’ = 0x65, ...
    - \n = newline character is represented by either one or two ASCII character

3.38 ASCII Table

<table>
<thead>
<tr>
<th>LSD/MSD</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NULL</td>
<td>DLW</td>
<td>SPACE</td>
<td>0</td>
<td>@</td>
<td>P</td>
<td>'</td>
<td>p</td>
</tr>
<tr>
<td>1</td>
<td>SOH</td>
<td>DC1</td>
<td>!</td>
<td>1</td>
<td>A</td>
<td>Q</td>
<td>a</td>
<td>q</td>
</tr>
<tr>
<td>2</td>
<td>STX</td>
<td>DC2</td>
<td>&quot;</td>
<td>2</td>
<td>B</td>
<td>R</td>
<td>b</td>
<td>r</td>
</tr>
<tr>
<td>3</td>
<td>ETX</td>
<td>DC3</td>
<td>#</td>
<td>3</td>
<td>C</td>
<td>S</td>
<td>c</td>
<td>s</td>
</tr>
<tr>
<td>4</td>
<td>EOT</td>
<td>DC4</td>
<td>$</td>
<td>4</td>
<td>D</td>
<td>T</td>
<td>d</td>
<td>t</td>
</tr>
<tr>
<td>5</td>
<td>ENQ</td>
<td>NAK</td>
<td>%</td>
<td>5</td>
<td>E</td>
<td>U</td>
<td>e</td>
<td>u</td>
</tr>
<tr>
<td>6</td>
<td>ACK</td>
<td>SYN</td>
<td>&amp;</td>
<td>6</td>
<td>F</td>
<td>V</td>
<td>f</td>
<td>v</td>
</tr>
<tr>
<td>7</td>
<td>BEL</td>
<td>ETB</td>
<td>‘</td>
<td>7</td>
<td>G</td>
<td>W</td>
<td>g</td>
<td>w</td>
</tr>
<tr>
<td>8</td>
<td>BS</td>
<td>CAN</td>
<td>{</td>
<td>8</td>
<td>H</td>
<td>X</td>
<td>h</td>
<td>x</td>
</tr>
<tr>
<td>9</td>
<td>TAB</td>
<td>EM</td>
<td></td>
<td>9</td>
<td>I</td>
<td>Y</td>
<td>i</td>
<td>y</td>
</tr>
<tr>
<td>A</td>
<td>LF</td>
<td>SUB</td>
<td>:</td>
<td>J</td>
<td>Z</td>
<td>x</td>
<td>j</td>
<td>z</td>
</tr>
<tr>
<td>B</td>
<td>VT</td>
<td>ESC</td>
<td>+</td>
<td>K</td>
<td>[</td>
<td>k</td>
<td>{</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>FF</td>
<td>FS</td>
<td>&lt;</td>
<td>L</td>
<td>\</td>
<td>l</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>CR</td>
<td>GS</td>
<td>-</td>
<td>M</td>
<td>]</td>
<td>m</td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>SO</td>
<td>RS</td>
<td>&gt;</td>
<td>N</td>
<td>^</td>
<td>n</td>
<td>~</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>SI</td>
<td>US</td>
<td>/</td>
<td>O</td>
<td>_</td>
<td>o</td>
<td>DEL</td>
<td></td>
</tr>
</tbody>
</table>

3.39 UniCode
- ASCII can represent only the English alphabet, decimal digits, and punctuation
  - 7-bit code => 2^7 = _____ characters
  - It would be nice to have one code that represented more alphabets/characters for common languages used around the world
- Unicode
  - 16-bit Code => ______ characters
  - Represents many languages alphabets and characters
  - Used by Java as standard character code

3.40 Summary
- Convert Base r to Base 10
  - Apply place values (powers of r)
    - \(N_r \rightarrow \sum (a_i \times r^i) \rightarrow D_{10}\)
- Convert Base 10 to Base r
  - "Make change" using powers of r as the weights/denominations
- Base 2 (Bin) ↔ Base 16 (Hex)
  - Group or expand 1 hex digit to/from 4 bits
  - Start at binary point and work outward