Binary Representation

Claude Shannon

- Claude Shannon was one of the pioneers who shaped computer science in its early years. In his master's thesis, Shannon showed how it was possible to use Boolean logic and switching circuits to perform arithmetic calculations. That work led directly to the use of binary arithmetic inside computers, an idea that at first generated considerable resistance among more established scientists.

- Shannon is best known as the inventor of information theory, which provides the mathematical foundation for understanding data communication. To many, however, Shannon's great contribution was that he never lost his sense of fun throughout his long career.

The Power of Bits

- The fundamental unit of memory inside a computer is called a bit—a term introduced in a paper by Claude Shannon as a contraction of the words binary digit.

- An individual bit exists in one of two states, usually denoted as 0 and 1.

- More sophisticated data can be represented by combining larger numbers of bits:
  - Two bits can represent four \((2 \times 2)\) values.
  - Three bits can represent eight \((2 \times 2 \times 2)\) values.
  - Four bits can represent 16 \((2^4)\) values, and so on.

- This laptop has 16GB of main memory and can therefore exist in \(2^{32}\) states. If you were to write that number out, it would contain more than fifty billion digits.

Leibniz and Binary Notation

- Binary notation is an old idea. It was described back in 1703 by the German mathematician Gottfried Wilhelm von Leibniz.

- Writing in the proceedings of the French Royal Academy of Science, Leibniz describes his use of binary notation in a simple, easy-to-follow style.

- Leibniz's paper further suggests that the Chinese were clearly familiar with binary arithmetic 2000 years earlier, as evidenced by the patterns of lines found in the I Ching.

Using Bits to Represent Integers

- Binary notation is similar to decimal notation but uses a different base. Decimal numbers use 10 as their base, which means that each digit counts for ten times as much as the digit to its right. Binary notation uses base 2, which means that each position counts for twice as much, as follows:

  0 0 1 1 0 1 0 1

  0 x 1 = 0
  1 x 2 = 2
  1 x 4 = 0
  1 x 8 = 8
  1 x 16 = 16
  0 x 32 = 32
  0 x 64 = 0
  0 x 128 = 0

  42

- Numbers and Bases

  - The calculation at the end of the preceding slide makes it clear that the binary representation 00101010 is equivalent to the number 42. When it is important to distinguish the base, the text uses a small subscript, like this: \(00101010\) is 42. Although it is useful to be able to convert a number from one base to another, it is important to remember that the number remains the same. What changes is how you write it down.

  - The number 42 is what you get if you count how many stars are in the pattern at the right. The number is the same whether you write it in English as forty-two, in decimal as 42, or in binary as 00101010.

  - Numbers do not have bases; representations do.
Octal and Hexadecimal Notation

- Because binary notation tends to get rather long, computer scientists often prefer octal (base 8) or hexadecimal (base 16) notation instead. Octal notation uses eight digits: 0 to 7. Hexadecimal notation uses sixteen digits: 0 to 9, followed by the letters A through F to indicate the values 10 to 15.
- The following diagrams show how the number forty-two appears in both octal and hexadecimal notation:

\[
\begin{align*}
\text{Octal} &: \quad 2 \times 8^1 + 5 \times 8^0 = 21 \quad \text{or} \quad 25_{8} \\
\text{Hexadecimal} &: \quad 1 \times 16^1 + 2 \times 16^0 = 26 \quad \text{or} \quad 1A_{16}
\end{align*}
\]

- The advantage of using either octal or hexadecimal notation is that doing so makes it easy to translate the number back to individual bits because you can convert each digit separately.

Exercises: Number Bases

- What is the decimal value for each of the following numbers?
  - 10001_2
  - 177_8
  - AD_{16}
- Every binary file produced by the Java compiler begins with the following sixteen bits:
  \[
  110010101011111110
  \]
  How would you express that number in hexadecimal notation?

Binary Arithmetic and Hardware

- Binary arithmetic is particularly suitable for computing hardware because the digits 0 and 1 correspond closely to the “off” and “on” states of an electrical circuit.
- As Claude Shannon showed in his Master’s thesis in 1937, you can use switching circuits to implement all the operations of binary arithmetic by defining the absence of an electrical connection to be the digit 0 and the presence of a connection to be the digit 1.

The AND Circuit

\[
\begin{align*}
\text{Input} &: \quad 0 \quad \text{or} \quad 1 \\
\text{Output} &: \quad 0 \quad \text{if both inputs are 0, \quad 1 otherwise}
\end{align*}
\]

The OR Circuit

\[
\begin{align*}
\text{Input} &: \quad 0 \quad \text{or} \quad 1 \\
\text{Output} &: \quad 1 \quad \text{if at least one input is 1, \quad 0 otherwise}
\end{align*}
\]

Implementing NOT with Switches

\[
\begin{align*}
\text{Input} &: \quad 0 \quad \text{or} \quad 1 \\
\text{Output} &: \quad 1 \quad \text{if the input is 0, \quad 0 otherwise}
\end{align*}
\]
Exercise: Controlling Room Lights

- Suppose you have a room with one light that you want to control from two switches at opposite ends of the room. How would you implement this functionality with switches?

The XOR Circuit

The XOR Circuit

The XOR Circuit

The XOR Circuit

Bits and Representation

- Sequences of bits have no intrinsic meaning except for the representation that we assign to them, both by convention and by building particular operations into the hardware.
- As an example, a 32-bit word represents an integer only because we have designed hardware that can manipulate those words arithmetically, applying operations such as addition, subtraction, and comparison.
- By choosing an appropriate representation, you can use bits to represent any value you can imagine:
  - Characters are represented using numeric character codes.
  - Floating-point representation supports real numbers.
  - Two-dimensional arrays of bits represent images.
  - Sequences of images represent video.
  - And so on . . .
Bits Are Everywhere

The ASCII Subset of Unicode

- The following table shows the first 128 characters in the Unicode character set, which are the same as in the older ASCII scheme:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>U0</td>
<td>V0</td>
<td>V1</td>
<td>V2</td>
<td>V3</td>
<td>V4</td>
<td>V5</td>
<td>V6</td>
<td>V7</td>
<td>V8</td>
<td>V9</td>
<td>VA</td>
<td>VB</td>
<td>VC</td>
<td>VD</td>
<td>VE</td>
</tr>
<tr>
<td>02</td>
<td>03</td>
<td>04</td>
<td>05</td>
<td>06</td>
<td>07</td>
<td>08</td>
<td>09</td>
<td>0A</td>
<td>0B</td>
<td>0C</td>
<td>0D</td>
<td>0E</td>
<td>0F</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>1A</td>
<td>1B</td>
<td>1C</td>
<td>1D</td>
<td>1E</td>
<td>1F</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>2A</td>
<td>2B</td>
<td>2C</td>
<td>2D</td>
<td>2E</td>
<td>2F</td>
<td>30</td>
<td>31</td>
</tr>
</tbody>
</table>

- The table shows, for example, that the ASCII character “*” has the hexadecimal value 2A, which is 42 in decimal.

Representing Characters

- Computers use numeric encodings to represent character data inside the memory of the machine, in which each character is assigned an integral value.

- Character codes, however, are not very useful unless they are standardized. When different computer manufacturers use different coding sequence (as was indeed the case in the early years), it is harder to share such data across machines.

- The first widely adopted character encoding was ASCII (American Standard Code for Information Interchange).

- With only 256 possible characters, the ASCII system proved inadequate to represent the many alphabets in use throughout the world. It has therefore been superseded by Unicode, which allows for a much larger number of characters.

The chr and ord Functions

- Python includes two built-in functions to simplify conversion between an integer and the corresponding Unicode character.

  - The `chr` function takes an integer and returns a one-character string containing the Unicode character with that code.
    - `chr(32)` ⇒ “ “ (the space character)
    - `chr(65)` ⇒ “A” (an uppercase A)
    - `chr(960)` ⇒ “π” (the Greek letter pi)

  - The `ord` function takes a one-character string and returns the value of that character in Unicode.
    - `ord(" ")` ⇒ 32
    - `ord("A")` ⇒ 65
    - `ord("π")` ⇒ 960

Other Unicode Blocks

- The Greek alphabet appears in the block beginning at 390.

- The Unicode block beginning at 2630 includes symbols from the I Ching, astronomy, astrology, games, and music.

Hardware Support for Characters