Representing Letters: ASCII
Representing numbers is great -- but what about words? Can we make sentences with binary data?

- **Key Idea:** Every letter is ________ binary bits. "*: in ASCII
  (This means that every letter is _______ hex digits.)

- Global standard called the American Standard Code for Information Interchange (ASCII) is a ___________ for translating numbers to characters.

<table>
<thead>
<tr>
<th>ASCII Character Encoding Examples:</th>
<th>Binary</th>
<th>Hex</th>
<th>Char.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b 0100 0001</td>
<td>0x41</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>0b 0100 0010</td>
<td>0x42</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>0b0010 0100 0x24</td>
<td>$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...and now we can form sentences!

Q: Are there going to be any issues with ASCII?

Representing Letters: Other Character Encodings
Since ASCII uses only 8 bits, we are limited to only 256 unique characters. There’s far more than 256 characters -- and what about EMOJIs?? 🎉

- **Many** other character encodings exist other than ASCII.
- The most widely used character encoding is known as **Unicode Transformation Format (8-bit)** or ________.
- Standard is ISO/IEC **10646** (Updated annually!).

Technical Details of UTF-8 Encoding
UTF-8 uses a ___________ -bit design where each character becomes any of the following:

<table>
<thead>
<tr>
<th>Length</th>
<th>Byte #1</th>
<th>Byte #2</th>
<th>Byte #3</th>
<th>Byte #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-byte</td>
<td>0___</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-bytes:</td>
<td>110_ ___</td>
<td>10__</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-bytes:</td>
<td>1110 ___</td>
<td>10__</td>
<td>10__</td>
<td></td>
</tr>
<tr>
<td>4-bytes:</td>
<td>1111 0___</td>
<td>10__</td>
<td>10__</td>
<td>10__</td>
</tr>
</tbody>
</table>

Unicode characters are represented by **U+###** *(where ### is the hex value of the character encoding data)* and all 1-byte characters match the ASCII character encoding:

- ‘a’ is ASCII __________, or __________.

**Example:** ε (epsilon) is defined as **U+03b5**. How do we encode this?

**Example:** I received the following binary message encoded in UTF-8:

```
0100 1000 0110 1001 1111 0000 1001 1111 1000 1110 1000 1001
```

1. What is the hexadecimal representation of this message?

2. What is the byte length of this message? ________

3. What is the character length of this message? ________

4. What does the message say?
Bit Manipulation: Binary Addition
For the past two lectures we have focused on the first foundation: DATA. Today, we are going to begin the transition away from data and into how data applies to the CPU. Binary addition work just like decimal addition, but with only 0s and 1s:

\[
\begin{array}{c}
0b \ 010011 \\
+ \ 0b \ 001001 \\
\hline
0b \ 001110
\end{array}
\]

\[
\begin{array}{c}
0b \ 0011 \\
+ \ 0b \ 0111 \\
\hline
0b \ 1010
\end{array}
\]

Negative Numbers: _______________________________

\[
\begin{array}{c}
0b \ 010011 \\
- \ 0b \ 001001 \\
\hline
0b \ 001100
\end{array}
\]

\[
\begin{array}{c}
0b \ 0011 \\
- \ 0b \ 0111 \\
\hline
0b \ 1100
\end{array}
\]

Two’s Complement
The Two’s Complement is a way to represent signed (ex: positive vs. negative) numbers in a way ____________________________!

Overflow Detection in Two’s Complement:

Towards Multiplication
With Two’s Complement, we can add and subtract numbers! What about more complex operations?

For simplicity, let's imagine running on an 7-bit machine:

\[-17 = \]

\[-4 = \]

\[-1 = \]

Bit Shift Operations:

1. [Left Shift]:

2. [Right Shift]: