Representing Data: Hexadecimal
Binary data gets really long, really fast! The number of students enrolled at University of Illinois is \(0b1100\ 1100\ 0110\ 1011\)
- To represent binary data in a compact way, we often will use hexadecimal -- or “base-16” -- denoted by the prefix \(0x\).

Hexadecimal Digits:

<table>
<thead>
<tr>
<th>Place of Hexadecimal Numbers:</th>
<th>Hex Number:</th>
<th>Place Value:</th>
<th>Decimal Place Value:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>c</td>
<td>(16^5)</td>
<td>1048576</td>
</tr>
<tr>
<td></td>
<td>(\theta)</td>
<td>(16^4)</td>
<td>65536</td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>(16^3)</td>
<td>4096</td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>(16^2)</td>
<td>256</td>
</tr>
<tr>
<td></td>
<td>e</td>
<td>(16^1)</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>e</td>
<td>(16^0)</td>
<td>1</td>
</tr>
</tbody>
</table>

Translation from Decimal to Hexadecimal:

\[11_{10} = 0x\] \[87_{10} = 0x\] \[34_{10} = 0x\] \[255_{10} = 0x\]

Hexadecimal is particularly useful as it ________________:

University of Illinois student population in Fall 2020 (52,331):

\[0b\] 1100 1100 0110 1011
\[0x\]

Number of people following Taylor Swift on Twitter (88,681,056):

\[0b\] 101 0100 1001 0010 1010 0110 0000
\[0x\]

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.*
  
  (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.
- ASCII was not the first but was developed from a long history of other encodings. Charles MacKenzie’s “Coded Character Set, History, and Development” has an over 500-page history on character encodings! (Linked on the website!)

### ASCII Character Encoding Examples:

<table>
<thead>
<tr>
<th>Binary</th>
<th>Hex</th>
<th>Char.</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>0b 0110 0001</td>
<td>0x61</td>
<td>a</td>
</tr>
<tr>
<td>0b 0100 0010</td>
<td>0x42</td>
<td>B</td>
<td>0b 0110 0010</td>
<td>0x62</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C</td>
<td></td>
<td></td>
<td>c</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D</td>
<td></td>
<td></td>
<td>d</td>
</tr>
<tr>
<td>0b0010 0100 0010</td>
<td>0x24</td>
<td>$</td>
<td>0b0111 1011</td>
<td>0x7b</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 (Latest update is :2002, or v13).
UTF-8 uses a ___________-bit design where each character be 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 1000 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?

### Programming in C
One example of a plain-text file in a C source code file. Today, you’ll see your very first Machine Problem in CS 240!
- You already know how to program in C++! 🎉
- Programming in C is a simplification of the C++ programming.

1. Program Starting Point of ALL C PROGRAMS:

2. Printing Using printf() (from <stdio.h>):

```c
#include <stdio.h>

int main() {
    char *s = "Hello, world!";
    float f = 3.14;

    printf("%d %s %f
", i, s, f);
    printf("%d
", s[0]);
    printf("%f\n", s[0]);
    printf("%d\n", s);

    return 0;
}
```

printf has a variable number of arguments:
- First argument
- Additional arguments

3. Pointers:

```c
int main() {
    char *s = malloc(10);
    int *num = malloc(sizeof(int));

    printf("%p %p\n", s, num);
    return 0;
}
```

4. Heap Memory Allocation:

```c
int main() {
    char *s = malloc(10);
    int *num = malloc(sizeof(int));

    printf("%p %p\n", s, num);
    return 0;
}
```

5. Strings -- #include <string.h>
Four Key Functions:
- `strcmp(char *s1, char *s2)` -- Compares two strings
- `strcat(char *dest, char *src)` -- Concatenate two strings
- `strcpy(char *dest, char *src)` -- Copies a string
- `strlen(char *s)` -- Returns the length of the string