Who are You?
What did I miss?
Syllabus

• What are the goals of this class?
Overflow

• Number
Overflow

• Number
• Buffer
  – Smashing the stack for fun and profit
  – Phrak 49 (www.phrak.com)
  – Generally from strcpy(), strcat(), and sprintf()
  – Overflow fixed size buffer with machine code
**Bits**

**Electronic Implementation**

- Easy to store with bistable elements
- Reliably transmitted on noisy and inaccurate wires

![Diagram showing voltage levels for bits 0 and 1](image-url)
Bits + Context

• Integer
• Float
• Character
• Program instruction
• Memory Address

• *Bytes*
Background: Hexadecimal

- 0xA4
  - 10100100 (binary)
  - 164 (decimal)

<table>
<thead>
<tr>
<th>Hex</th>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0010</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0011</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0100</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0101</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0110</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>0111</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>1001</td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td>1010</td>
</tr>
<tr>
<td>B</td>
<td>11</td>
<td>1011</td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>1100</td>
</tr>
<tr>
<td>D</td>
<td>13</td>
<td>1101</td>
</tr>
<tr>
<td>E</td>
<td>14</td>
<td>1110</td>
</tr>
<tr>
<td>F</td>
<td>15</td>
<td>1111</td>
</tr>
</tbody>
</table>
Take out some paper...

- Convert 0x173A4C to binary
- Convert 11 1100 1010 1101 1011 0011_2 to hex
- Convert 7AF_{16} to base-10
- Convert 60_{10} to base-16
Byte-Oriented Memory Organization

Programs Refer to Virtual Addresses

- Conceptually very large array of bytes
- Actually implemented with hierarchy of different memory types
- System provides address space private to particular “process”
  - Program being executed
  - Program can clobber its own data, but not that of others

Compiler + Run-Time System Control Allocation

- Where different program objects should be stored
- All allocation within single virtual address space
Machine Words

Machine Has “Word Size”

- Nominal size of integer-valued data
  - Including addresses

- Most current machines use 32 bits (4 bytes) words
  - Limits addresses to 4GB
  - Becoming too small for memory-intensive applications

- High-end systems use 64 bits (8 bytes) words
  - Potential address space $\approx 1.8 \times 10^{19}$ bytes
  - x86-64 machines support 48-bit addresses: 256 Terabytes

- Machines support multiple data formats
  - Fractions or multiples of word size
  - Always integral number of bytes