Endianness
**Activities**

**This Week**
- Network programming
- Endianness
- Lab 8 – Network Programming

**Assignments Due**
- Lab 8
  - Due by Mar 27\(^{th}\) 5:00am
- Lab 9
  - Due by Apr 3\(^{rd}\) 5:00am
In typical computer memory, each address (location) stores one byte.

If we have a one-byte integer, how is that stored in memory?

If we have a two-byte integer, how is that stored in memory?

If we have a four-byte integer, how is that stored in memory?

Endianness = Byte Ordering
Endianness Example

32-bit hexadecimal number 0x12345678

Composed of 4 bytes:
0x12 0x34 0x56 0x78
(MSB) (LSB)

Two possible arrangements:

<table>
<thead>
<tr>
<th>Address</th>
<th>“Option A”</th>
<th>“Option B”</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x12</td>
<td>0x78</td>
</tr>
<tr>
<td>1</td>
<td>0x34</td>
<td>0x56</td>
</tr>
<tr>
<td>2</td>
<td>0x56</td>
<td>0x34</td>
</tr>
<tr>
<td>3</td>
<td>0x78</td>
<td>0x12</td>
</tr>
</tbody>
</table>
32-bit hexadecimal number
0x12345678

Composed of 4 bytes:
0x12 0x34 0x56 0x78

(MSB) (LSB)

Two possible arrangements:
- **Big Endian**
- **Little Endian**

<table>
<thead>
<tr>
<th>Address</th>
<th>Big Endian</th>
<th>Little Endian</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x12 (MSB)</td>
<td>0x78 (LSB)</td>
</tr>
<tr>
<td>1</td>
<td>0x34</td>
<td>0x56</td>
</tr>
<tr>
<td>2</td>
<td>0x56</td>
<td>0x34</td>
</tr>
<tr>
<td>3</td>
<td>0x78</td>
<td>0x12</td>
</tr>
</tbody>
</table>
How is $\text{DEADBEEF}_{16}$ stored in little and big endian formats at address $21\text{C}_{16}$?

- Little endian
  - $21\text{C}_{16} = \text{EF}_{16}$
  - $21\text{D}_{16} = \text{BE}_{16}$
  - $21\text{E}_{16} = \text{AD}_{16}$
  - $21\text{F}_{16} = \text{DE}_{16}$

- Big endian
  - $21\text{C}_{16} = \text{DE}_{16}$
  - $21\text{D}_{16} = \text{AD}_{16}$
  - $21\text{E}_{16} = \text{BE}_{16}$
  - $21\text{F}_{16} = \text{EF}_{16}$
Big Endian – vs – Little Endian

**Big-Endian CPU**

- Most significant byte (MSB) comes first (stored in lower memory address)

**Examples**
- Motorola 68000
- Java virtual machine
- IBM PowerPC (by default, can also be little endian)

**Little-Endian CPU**

- Least significant byte (LSB) comes first (stored in lower memory addresses)

**Examples**
- Intel x86/x86-64
- DEC Alpha
- ARM (by default, also can be big endian)
Origin in 1980s

Reference to Swift's *Gulliver's Travels*, in which the Lilliputians were divided into two camps:

- Those who ate their eggs by opening the ‘big’ end
- Those who ate them by opening the ‘little’ end

In other words, a trivial distinction
When do I need to care that some computers are big-endian and others are little endian?

What happens if I open big-endian data on a little-endian computer?

Endianness must be considered whenever you are sharing data between different computer systems.

- Reading/writing data files to disk
- Reading/writing data files to network
Best Practices

- Pick one format and stick with it!
  - Example: Data sent over the network will always be in \textit{big-endian} format regardless of who sends it
  - Networks are \textit{big-endian} “by tradition”
  - Example: Data written to disk will always be in \textit{little-endian} format regardless of who writes it

- Convert between data storage/transfer format and internal representation as needed
  - Example: Little-endian machines convert to big-endian before sending data onto the network (and convert back upon receiving data from the network)
### Examples in Industry

#### Table 2- Common file formats

<table>
<thead>
<tr>
<th>Little-Endian Format</th>
<th>Big-Endian Format</th>
<th>Variable or Bi-Endian Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMP (Windows* &amp; OS/2)</td>
<td>PSD (Adobe Photoshop*)</td>
<td>DXF (AutoCAD*)</td>
</tr>
<tr>
<td>GIF</td>
<td>IMG (GEM Raster*)</td>
<td>PS (Postscript*, 8 bit interpreted text, no Endian issue)</td>
</tr>
<tr>
<td>FLI (Autodesk Animator*)</td>
<td>JPEG, JPG</td>
<td>POV (Persistence of Visionraytracer*)</td>
</tr>
<tr>
<td>PCX (PC Paintbrush*)</td>
<td>MacPaint</td>
<td>RIFF (WAV &amp; AVI*)</td>
</tr>
<tr>
<td>QTM (MAC Quicktime*)</td>
<td>SGI (Silicon Graphics*)</td>
<td>TIFF</td>
</tr>
<tr>
<td>RTF (Rich Text Format)</td>
<td>Sun Raster</td>
<td>XWD (X Window Dump*)</td>
</tr>
<tr>
<td>WPG (WordPerfect*)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bus Protocols</th>
<th>Network Protocols</th>
<th>Bus Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infiniband</td>
<td>TCP/IP</td>
<td>GMII (8 bit wide bus, no Endian issue)</td>
</tr>
<tr>
<td>PCI Express</td>
<td>UDP</td>
<td></td>
</tr>
<tr>
<td>PCI-32/PCI-64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USB</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>