**Number Representations**

### Integer Numbers

- Integers are almost universally stored as binary numbers.
- When you enter an integer from the keyboard, software converts the ASCII or Unicode characters to a binary integer.

### Decimal Numbers

- Some systems (e.g., Intel Pentium) support decimal numbers in packed or unpacked format.
  - **Packed decimal** uses 4 bit fields for each digit:
    - $9375 = 1001,0011,0111,0101$
  - **Unpacked decimal** uses a byte per digit (ASCII):
    - $9375 = 00111001,00110011,00110111,00110101$

### Negative Integers

- Almost all systems for storing negative binary numbers set the left most bit (MSB) to indicate the sign of a number.
- Common formats:
  - Signed Magnitude
  - Ones Complement
  - Twos Complement

### Signed Magnitude

- Negative numbers are the same as positive with the sign bit set:
- Normal addition does not work for negative numbers:
  - $001 + 101 = 110$
  - $1 + (-1) = -2$

### Ones Complement

- Negative number are the logical inverse of positive numbers:
- Mathematically positive and negative zero are the same, but they are different bit patterns.

### Ones Complement Arithmetic

- The carry out of the sign position needs to be added to the number:
  - $101$ + $-2$
  - $+110$ + $-1$
  - $011$
  - $+1$ add carry out of sign
  - $100$ -3
Twos Complement

- Negative numbers are the logical inverse of positive numbers plus 1.

<table>
<thead>
<tr>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>-4</td>
<td>-3</td>
<td>-2</td>
<td>-1</td>
</tr>
</tbody>
</table>

Three bit example

```
111 110 10 110 01 110
```

Normal binary arithmetic works for positive and negative numbers.

Byte Ordering

- Some systems store the least significant byte first (Little Endian). Others store the most significant byte first (Big Endian).
- The decimal number 258 (0100000010₂) would be stored in as a 16 bit binary number.
  - Big Endian: 00000001 00000010
  - Little Endian: 00000010 00000001

Floating Point

- Almost all systems today use the IEEE 754 standard for floating point numbers.
- The standard supports 32 bit (single precision) and 64 bit (double precision).
- Standard committee is working on 128 bit
- In many engineering applications, accuracy is extremely important.

Floating Point Format

<table>
<thead>
<tr>
<th>Single Precision up to 2.0x10^{38}</th>
</tr>
</thead>
<tbody>
<tr>
<td>sign</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Double Precision up to 2.0x10^{308}</th>
</tr>
</thead>
<tbody>
<tr>
<td>sign</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

Normalization

- The mantissa or fraction part of a floating point number is normalized. This means that the fraction is shifted and the exponent adjusted so that the left most bit is a one.
- Since the left most bit is always a one, it is not stored in the number. It is assumed to always be there.

Special Values

<table>
<thead>
<tr>
<th>Exponent</th>
<th>Mantissa</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Zero</td>
</tr>
<tr>
<td>255</td>
<td>0</td>
<td>±Infinity</td>
</tr>
<tr>
<td>255</td>
<td>nonzero</td>
<td>NaN not a number</td>
</tr>
</tbody>
</table>