Shift operations

- **Shifting bit patterns**
  - What if shift > sizeof variable type? \(\leftarrow\) “undefined”
    - “warning: right shift count >= width of type”
    - \(k \text{ (shift)} \mod w \text{ (width)}\)
  - What if shift using a negative number? \(\leftarrow\) “undefined”
    - “warning: right shift count is negative”
    - \(w \text{ (width)} - k \text{ (neg shift)}\)
- **Left shift**
  - Always brings in zero, but
    - What if left shift a signed value? Oh well...
- **Right shift**
  - Logical – is unsigned; always brings in zeros
  - Arithmetic – is signed; repeats left most bit (sign bit)
    - sign specification default in C is “signed”
    - Almost all compilers/machines do repeat sign (most)

- **Examples** (shiftex.c)
  - Note: difference between 0xF0F0 and 0xF0F0F0F0F0?
Logical operations

- treat any nonzero argument as representing TRUE and zero as representing FALSE
- RETURN either 0 (false) or 1 (true)

Difference between bit and logical

- & and &&
- | and ||
- ~ and !
Different encoding scheme than float
The left-most bit is the sign bit if using a signed data type.
* Total number of values: $2^w$

Unsigned $\rightarrow$ non-neg numbers ($\geq 0$)
  * Minimum value: 0
  * Maximum value: $2^w - 1$

Signed $\rightarrow$ neg, zero, and pos numbers
  * Minimum value: $-2^{w-1}$
  * Maximum value: $2^{w-1} - 1$

* Where $w$ is the bit width of the data type
Integer Decoding

- Binary to Decimal (mult of powers)
- Unsigned = simple binary = B2U
  - You already know how to do this :o)
  - 0101 = 5, 1111 = F, 1110 = E, 1001 = 9
- Signed = two’s complement = B2T*
  - 0 101 = unsigned = 5
  - 1 111 = -1*2^3 + 7 = -8 + 7 = -1
  - 1 110 = -1*2^3 + 6 = -8 + 6 = -2
  - 1 001 = -1*2^3 + 1 = -8 + 1 = -7
  - Another way, if sign bit = 1
    - invert bits and add 1
    - right to left, leave alone to first 1, then invert rest
    - 1010...

* reminder: left most bit is sign bit
B2O & B2S

- One’s complement = bit complement of B2U
- Signed Magnitude = left most bit set to 1 with B2U for the remaining bits
- Both include neg values
- Min/max = \(-2^{w-1}-1\) to \(2^{w-1}-1\)
- Pos and neg zero
- Difficulties with arithmetic options

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
<td>-8</td>
<td>-7</td>
<td>-0</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>-7</td>
<td>-6</td>
<td>-1</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
<td>-6</td>
<td>-5</td>
<td>-2</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
<td>-5</td>
<td>-4</td>
<td>-3</td>
</tr>
<tr>
<td>1100</td>
<td>12</td>
<td>-4</td>
<td>-3</td>
<td>-4</td>
</tr>
<tr>
<td>1101</td>
<td>13</td>
<td>-3</td>
<td>-2</td>
<td>-5</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
<td>-2</td>
<td>-1</td>
<td>-6</td>
</tr>
<tr>
<td>1111</td>
<td>15</td>
<td>-1</td>
<td>-0</td>
<td>-7</td>
</tr>
</tbody>
</table>