• Time in Tue evenings: 7-8pm

• $1111111111111111 = 2^{16}-1$

• Casting

• $x<<y+z<<3$
• 16 bits
• What’s the maximum UNSIGNED # possible?

\[ 1111 \ 1111 \ 1111 \ 1111 \]

\[ = 2^{16} - 1 \]

(Exponents go from 0-15, not 1-16)
Mapping Between Signed & Unsigned

**Two’s Complement**

\[ x \] \rightarrow \text{T2B} \rightarrow \text{T2U} \rightarrow \text{B2U} \rightarrow \text{UnSigned} \rightarrow \[ ux \]

*Maintain Same Bit Pattern*

**Unsigned**

\[ ux \] \rightarrow \text{U2B} \rightarrow \text{U2T} \rightarrow \text{B2T} \rightarrow \text{Two’s Complement} \rightarrow \[ x \]

*Maintain Same Bit Pattern*

- Mappings between unsigned and two’s complement numbers: keep bit representations and reinterpret
### Mapping Signed ↔ Unsigned

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

= +16
Relation between Signed & Unsigned

Two’s Complement

\[ x \xrightarrow{T2B} X \xrightarrow{B2U} ux \]

Maintain Same Bit Pattern

Unsigned

\[ ux = \begin{cases} 
  x & x \geq 0 \\
  x + 2^w & x < 0 
\end{cases} \]

Large negative weight becomes Large positive weight
Conversion Visualized

- 2’s Comp. → Unsigned
  - Ordering Inversion
  - Negative → Big Positive

2’s Complement Range

2’s Complement Range

Unsigned Range

$T_{Max}$

$0$

$-1$

$-2$

$T_{Min}$

$T_{Max}$

$U_{Max}$

$U_{Max} - 1$

$T_{Max} + 1$

$0$
Signed vs. Unsigned in C

- **Constants**
  - By default are considered to be signed integers
  - Unsigned if have “U” as suffix
    
    `0U, 4294967259U`

- **Casting**
  - Explicit casting between signed & unsigned same as U2T and T2U
    
    ```c
    int tx, ty;
    unsigned ux, uy;
    tx = (int) ux;
    uy = (unsigned) ty;
    ```

  - Implicit casting also occurs via assignments and procedure calls
    
    ```c
    tx = ux;
    uy = ty;
    ```
2’s compliment trick

\(\sim x + 1 = \) ?????

Why?
Negation: Complement & Increment

- **Claim:** Following holds for 2’s Complement
  \[ \sim x + 1 = -x \]

- **Complement**
  - **Observation:** \( \sim x + x = 1111...111 = -1 \)
    
    \[
    \begin{array}{c}
    x \quad 100111101 \\
    \sim x \quad 01100010 \\
    \hline
    \text{111111111}
    \end{array}
    \]

- **Complete Proof?**
Casting Surprises

Expression Evaluation
- If mix unsigned and signed in single expression, *signed values implicitly cast to unsigned*
- Including comparison operations <, >, ==, <=, >=
- Examples for $W = 32$: $T_{MIN} = -2,147,483,648$, $T_{MAX} = 2,147,483,647$

<table>
<thead>
<tr>
<th>Constant$_1$</th>
<th>Constant$_2$</th>
<th>Relation</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>0U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2147483647</td>
<td>-2147483647-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2147483647U</td>
<td>-2147483647-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(unsigned)-1</td>
<td>-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2147483647</td>
<td>2147483648U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2147483647</td>
<td>(int) 2147483648U</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Casting Surprises

Expression Evaluation

- If mix unsigned and signed in single expression, *signed values implicitly cast to unsigned*
- Including comparison operations `<`, `>`, `==`, `<=`, `>=`
- Examples for W = 32:  $T_{MIN} = -2,147,483,648$,  $T_{MAX} = 2,147,483,647$

<table>
<thead>
<tr>
<th>Constant₁</th>
<th>Constant₂</th>
<th>Relation</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0U</td>
<td>==</td>
<td>unsigned</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>&lt;</td>
<td>signed</td>
</tr>
<tr>
<td>-1</td>
<td>0U</td>
<td>&gt;</td>
<td>unsigned</td>
</tr>
<tr>
<td>2147483647</td>
<td>-2147483647-1</td>
<td>&gt;</td>
<td>signed</td>
</tr>
<tr>
<td>2147483647U</td>
<td>-2147483647-1</td>
<td>&gt;</td>
<td>unsigned</td>
</tr>
<tr>
<td>-1</td>
<td>-2</td>
<td>&gt;</td>
<td>signed</td>
</tr>
<tr>
<td>(unsigned)-1</td>
<td>-2</td>
<td>&gt;</td>
<td>unsigned</td>
</tr>
<tr>
<td>2147483647</td>
<td>2147483648U</td>
<td>&lt;</td>
<td>unsigned</td>
</tr>
<tr>
<td>2147483647</td>
<td>(int) 2147483648U</td>
<td>&gt;</td>
<td>signed</td>
</tr>
</tbody>
</table>
Summary

Casting Signed $\leftrightarrow$ Unsigned: Basic Rules

- Bit pattern is maintained
- But reinterpreted
- Can have unexpected effects: adding or subtracting $2^w$

- Expression containing signed and unsigned int
  - int is cast to unsigned!!
Sign Extension

- **Task:**
  - Given \( w \)-bit signed integer \( x \)
  - Convert it to \( w+k \)-bit integer with same value

- **Rule:**
  - Make \( k \) copies of sign bit:
  - \( X' = x_{w-1}, \ldots, x_{w-1}, x_{w-1}, x_{w-2}, \ldots, x_0 \)
Sign Extension Example

```c
short int x = 15213;
int ix = (int) x;
short int y = -15213;
int iy = (int) y;
```

<table>
<thead>
<tr>
<th></th>
<th>Decimal</th>
<th>Hex</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>15213</td>
<td>3B 6D</td>
<td>00111011 01101101</td>
</tr>
<tr>
<td>ix</td>
<td>15213</td>
<td>00 00 3B 6D</td>
<td>00000000 00000000 00111011 01101101</td>
</tr>
<tr>
<td>y</td>
<td>-15213</td>
<td>C4 93</td>
<td>11000100 10010011</td>
</tr>
<tr>
<td>iy</td>
<td>-15213</td>
<td>FF FF C4 93</td>
<td>11111111 11111111 11000100 10010011</td>
</tr>
</tbody>
</table>

- Converting from smaller to larger integer data type
- C automatically performs sign extension
Summary:
Expanding, Truncating: Basic Rules

- Expanding (e.g., short int to int)
  - Unsigned: zeros added
  - Signed: sign extension
  - Both yield expected result

- Truncating (e.g., unsigned to unsigned short)
  - Unsigned/signed: bits are truncated
  - Result reinterpreted
  - Unsigned: mod operation
  - Signed: similar to mod
  - For small numbers yields expected behaviour