Lecture 2: Representing Integers

CS 105

January 27, 2020
Abstraction
The C Language

• Syntax like Java: declarations, if, while, return

• Data and execution model are “closer to the machine”
  • More power and flexibility
  • More ways to make mistakes
  • Sometimes confusing relationships
  • Pointers!!
Memory: A (very large) array of bytes

• An index into the array is an address, location, or pointer
  • Often expressed in hexadecimal

• We speak of the value in memory at an address
  • The value may be a single byte …
  • … or a multi-byte quantity starting at that address

• Larger words (32- or 64-bit) are stored in contiguous bytes
  • The address of a word is the address of its first byte
  • Successive addresses differ by word size
Representing Unsigned Integers

- Think of bits as the binary representation

\[
\text{UnsignedValue}(x) = \sum_{j=0}^{w-1} x_j \cdot 2^j
\]

- If you have \( w \) bits, what is the range?
Endianness

BIG ENDIAN - The way people always broke their eggs in the Lilliput land

LITTLE ENDIAN - The way the king then ordered the people to break their eggs
Unsigned Integers in C

- What about casting?
  - Casting from shorter to longer types preserves the value
  - Casting from longer to shorter types truncates the bits

- What about negative numbers?
Representing Signed Integers

- Option 1: sign-magnitude
  - One bit for sign; interpret rest as magnitude

- Option 2: excess-K
  - Choose a positive K in the middle of the unsigned range
  - \( \text{SignedValue}(w) = \text{UnsignedValue}(w) - K \)

- Option 3: one’s complement
  - Flip every bit to get the negation
Representing Signed Integers

• Option 4: two’s complement
  • Most commonly used
  • Like unsigned, except the high-order contribution is negative

\[
\text{SignedValue}(x) = -x_{w-1} \cdot 2^{w-1} + \sum_{j=0}^{w-2} x_j \cdot 2^j
\]

• Exercise: Assume C short (2 bytes)
  • What is the binary representation for 47?
  • What is the hex representation for 47?
  • What is the binary representation for -47?
  • What is the hex representation for -47
### Example: Three-bit integers

<table>
<thead>
<tr>
<th>unsigned</th>
<th>signed</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>7</td>
</tr>
<tr>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>−1</td>
<td>111</td>
</tr>
<tr>
<td>−2</td>
<td>110</td>
</tr>
<tr>
<td>−3</td>
<td>101</td>
</tr>
<tr>
<td>−4</td>
<td>100</td>
</tr>
</tbody>
</table>

- The high-order bit is the *sign bit*.
- The largest unsigned value is $11\ldots1$, $\text{UMax}$.
- The signed value for $-1$ is always $11\ldots1$.
- Signed values range between $\text{TMin}$ and $\text{TMax}$.

This representation of signed values is called *two’s complement*. 
Two’s Complement Signed Integers

- “Signed” does not mean “negative”

- High order bit is the *sign bit*
  - To negate, complement all the bits and add 1

- Arithmetic is the same as unsigned—same circuitry

- Error conditions and comparisons are different
## Important Signed Numbers

<table>
<thead>
<tr>
<th></th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMax</td>
<td>0x7F</td>
<td>0x7FFF</td>
<td>0x7FFFFFFF</td>
<td>0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF</td>
</tr>
<tr>
<td>Tmin</td>
<td>0x80</td>
<td>0x8000</td>
<td>0x80000000</td>
<td>0x8000000000000000</td>
</tr>
<tr>
<td>0</td>
<td>0x00</td>
<td>0x0000</td>
<td>0x00000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>-1</td>
<td>0xFF</td>
<td>0xFFFF</td>
<td>0xFFFFFFFF</td>
<td>0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF</td>
</tr>
</tbody>
</table>
Unsigned and Signed Integers

- Use $w$-bit words; $w$ can be 8, 16, 32, or 64
- The bit sequence $b_{w-1} \ldots b_1 b_0$ represents an integer

<table>
<thead>
<tr>
<th>Value</th>
<th>Unsigned</th>
<th>Signed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smallest</td>
<td>$\sum_{i=0}^{w-1} b_i 2^i$</td>
<td>$-b_{w-1}2^{w-1} + \sum_{i=0}^{w-2} b_i 2^i$</td>
</tr>
<tr>
<td>Largest</td>
<td>$2^w - 1$</td>
<td>$2^{w-1} - 1$</td>
</tr>
</tbody>
</table>

Important!! “signed” does not mean “negative.”
Casting between Numeric Types

- Casting from shorter to longer types preserves the value
- Casting from longer to shorter types truncates the bits
- Casting between signed/unsigned types preserves the bits (it just changes the interpretation)
Exercise: Numeric Data Representations

- Assume you have a machine with 6-bit integers/3-bit shorts
- Assume variables: `int x = -17; short sy = -3;`
- Complete the following table

<table>
<thead>
<tr>
<th>Expression</th>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>-6</td>
<td></td>
<td>101010</td>
</tr>
<tr>
<td>(unsigned int) x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(int) sy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tmax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tmin</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>