Integer Representation

CMPU 224 – Computer Organization
Jason Waterman
Learning Objectives

- How integers are represented in modern computer systems
Representing integers

- Given $n$ bits to store an integer, we can represent $2^n$ different values.
- If we just care about non-negative (aka unsigned) integers, we can easily store the values $0, 1, 2, \ldots, 2^n-1$.
  - E.g., for 4 bits:
    - $0x2 = 2$
    - $0xB = 11$
    - $0xF = 15 = 2^4 - 1$
Integer overflow

• With \( n \) bits, we can represent values \( 0, 1, 2, \ldots, 2^n - 1 \)

• Overflow occurs when we have a result that doesn’t fit in the \( n \) bits
  
  • E.g., using 4 bits: \( 0xF + 0x1 \)

\[
\begin{align*}
0xF &= \quad 1111 \\
0x1 &= \quad 0001 \\
\hline \\
0xF + 0x1 &= 0x0 \quad \text{Overflow!!}
\end{align*}
\]
Integer overflow
Integer overflow
Representing negative integers

- We have seen how to represent **unsigned integers** (i.e., non-negative integers) as **unsigned binary** numbers
  - Addition works as we expect it to
  - Every number between 0 and $2^w-1$ has a unique encoding as a $w$-bit value

- How do we represent negative integers?

- Three common encodings:
  - Sign and magnitude
  - Ones’ complement
  - Two’s complement
Sign and magnitude

- Use one bit to represent sign, remaining bits represent magnitude
- With \( n \) bits, have \( n-1 \) bits for magnitude
  - E.g., with 4 bits, can represent integers
    - -7, -6, ..., -1, 0, 1, ..., 6, 7

\[ 1011 \]

- sign: -
- magnitude: 3

represents -3
Properties of sign and magnitude

• Straight-forward and intuitive

• Two different representations of zero!
  • E.g., using 4 bits, 1000 and 0000 both represent zero!

• Arithmetic operations need different implementations for signed and unsigned numbers
  • E.g., addition, using 4 bits
    • unsigned: 0001 + 1001 = 1 + 9 = 10 = 1010
    • sign and magnitude: 0001 + 1001 = 1 + -1 = 0 = 0000
Ones’ complement

- Negation is performed by performing a bitwise not (~) on the number
  - In other words, flip all the bits in the number
  - For example, using 4 bit numbers:
    - 6 = 0110
    - -6 = 1001
  - Equivalent to flipping every bit of b

- Like sign and magnitude, first bit indicates whether number is negative
  - If the msb (most significant bit) is 0, treat it like an unsigned binary number
  - If the msb is 1, the number is negative, flip all the bits to see its magnitude

- Using n bits, can represent numbers $2^n - 1$ values
  - E.g., using 4 bits, can represent integers
    - -7, -6, ..., -1, 0, 1, ..., 6, 7
Ones’ Compliment Practice

- Try it. Fill in the decimal values for these 4-bit Ones’ Complement numbers

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>-7</td>
</tr>
<tr>
<td>1001</td>
<td>-6</td>
</tr>
<tr>
<td>1010</td>
<td>-5</td>
</tr>
<tr>
<td>1011</td>
<td>-4</td>
</tr>
<tr>
<td>1100</td>
<td>-3</td>
</tr>
<tr>
<td>1101</td>
<td>-2</td>
</tr>
<tr>
<td>1110</td>
<td>-1</td>
</tr>
<tr>
<td>1111</td>
<td>0</td>
</tr>
</tbody>
</table>

*Flip the bits To get the magnitude of negative numbers*
Properties of ones’ complement

• We still have two different representations of zero!
  • E.g., using 4 bits, 1111 and 0000 both represent zero!

• Same implementation of arithmetic operations for signed and unsigned!
  • E.g., addition using 4 bits
    • unsigned: 0001 + 1001 = 1 + 9 = 10 = 1010
    • ones’ complement: 0001 + 1001 = 1 + -6 = -5 = 1010
Two’s complement

• Take the ones’ complement of the number and add one
  • Flip all the bits and add one to the number

• Using n bits, can represent numbers $2^n$ values
  • E.g., using 4 bits, can represent integers
    -8, -7, ..., -1, 0, 1, ..., 6, 7
  • Like sign and magnitude and ones’ complement, first bit indicates whether number is negative
Two’s Compliment Practice

• Try it. Fill in the values for these 4-bit Two’s Complement numbers

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

• Same implementation of arithmetic operations for signed and unsigned
  • E.g., addition, using 4 bits
    • unsigned: 0001 + 1001 = 1 + 9 = 10 = 1010
    • two’s complement: 0001 + 1001 = 1 + -7 = -6 = 1010

• Only one representation of zero!
  • Simpler to implement operations

• Not symmetric around zero
  • Can represent one more negative number than positive numbers

• Most common representation of negative integers in computers
Converting to and from two’s complement

• To encode a negative number in two’s complement in n bits:
  • Compute out the binary notation for the absolute value using all n bits
  • Invert the bits and add 1
  • E.g., to encode -5 using 8 bits
    • 5 = 00000101 using all 8 bits
    • Invert the bits: 11111010
    • Add one: 11111010 + 1 = 11111011
    • -5 encoded in two’s complement using 8 bits is 11111011

• To decode two’s complement:
  • If the first bit is 0 then number is positive
  • If the first bit is 1, then number is negative:
    • invert bits and add 1
    • Gives you the magnitude of the number
Two’s Complement Interpretation

• You can interpret a two’s complement number as having a negative weight in the MSB

<table>
<thead>
<tr>
<th>-2^3 = -8</th>
<th>2^2 = 4</th>
<th>2^1 = 2</th>
<th>2^0 = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>-8</td>
<td>+4</td>
<td>+0</td>
<td>+1</td>
</tr>
</tbody>
</table>

• One cool trick for converting to two’s complement
  • Flip all the bits, then starting from the LSB, flip all the ones you see (to zero) until you get to a zero. Flip that zero (to a one) and stop.
  • 00100 (4)
  • 11011 (flip all the bits)
  • 11100 (flip the rightmost 1’s and the first 0)
Different representations of 4-bit numbers

<table>
<thead>
<tr>
<th>Binary</th>
<th>Unsigned</th>
<th>Sign&amp;Mag</th>
<th>Ones’ Comp</th>
<th>Two’s Comp</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>-0</td>
<td>-7</td>
<td>-8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>-1</td>
<td>-6</td>
<td>-7</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
<td>-2</td>
<td>-5</td>
<td>-6</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
<td>-3</td>
<td>-4</td>
<td>-5</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>-5</td>
<td>-2</td>
<td>-3</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
<td>-6</td>
<td>-1</td>
<td>-2</td>
</tr>
<tr>
<td>1111</td>
<td>15</td>
<td>-7</td>
<td>-0</td>
<td>-1</td>
</tr>
</tbody>
</table>
What is going on with this cartoon?

They are counting sheep using a 16-bit two’s complement number!