Integer Representation

Lecture: 3

CMPU 224 – Computer Organization
Jason Waterman
Learning Objectives

• How integers are represented in modern computer systems
Word-Oriented Memory Organization

- Address Specify Byte locations
  - Address of the first byte in word
  - Addresses of successive words differ by 4 (32-bit) or 8 (64-bit) bytes
Byte Ordering

- So, how are the bytes within a multi-byte word ordered in memory
- Conventions
  - Big Endian: Sun, PowerPC Macs, Internet
    - Least significant byte has the highest address
  - Little Endian: x86, ARM processors running Android, iOS, and Windows
    - Least significant byte has the lowest address

<table>
<thead>
<tr>
<th>32-bit Words</th>
<th>64-bit Words</th>
<th>Bytes</th>
<th>Addr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addr = 0000</td>
<td>Addr = 0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>Addr = 0004</td>
<td>Addr = 0004</td>
<td>0001</td>
<td>0001</td>
</tr>
<tr>
<td>Addr = 0008</td>
<td>Addr = 0008</td>
<td>0002</td>
<td>0002</td>
</tr>
<tr>
<td>Addr = 0012</td>
<td>Addr = 0012</td>
<td>0003</td>
<td>0003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0004</td>
<td>0004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0005</td>
<td>0005</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0006</td>
<td>0006</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0007</td>
<td>0007</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0008</td>
<td>0008</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0009</td>
<td>0009</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0010</td>
<td>0010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0011</td>
<td>0011</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0012</td>
<td>0012</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0013</td>
<td>0013</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0014</td>
<td>0014</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0015</td>
<td>0015</td>
</tr>
</tbody>
</table>
Byte Ordering Example

- Example
  - 4-byte value (say an integer) of 0x01234567 is located at memory address 0x100
  - This value exists in memory locations 0x100, 0x101, 0x102, 0x103

![Big Endian Diagram]

![Little Endian Diagram]
Binary Arithmetic

• Works just like decimal arithmetic!
• Example: $6 + 5 = 11$
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, ..., 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

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

\[ \text{Add}_4(u, v) \]

Integer Addition

Diagram showing the addition of two integers, \( u \) and \( v \), with a 3D graph illustrating the result of \( \text{Add}_4(u, v) \).
Integer overflow

![Graph of UAdd4(u, v) with overflow indicated.](image)
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 represents -3

- sign: -ve
- magnitude: 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

- If integer \( k \) is represented by bits \( b_{n-1}...b_0 \), then \(-k\) is represented by
  \( 11...11 - b_{n-1}...b_0 \) (where \(|11...11|=n\))
  - E.g., using \( n=4 \) bits:
    - \( 6 = 0110 \)
    - \(-6 = 1111 - 0110 = 1001 \)
    - Equivalent to flipping every bit of \( b \)
  - Using \( n \) bits, can represent numbers \( 2^n-1 \) values
    - E.g., using 4 bits, can represent integers
      - \(-7, -6, ..., -1, 0, 1, ..., 6, 7 \)
    - 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
Ones’ Compliment Practice

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

<table>
<thead>
<tr>
<th>0000</th>
<th>0001</th>
<th>0010</th>
<th>0011</th>
<th>0100</th>
<th>0101</th>
<th>0110</th>
<th>0111</th>
<th>1000</th>
<th>1001</th>
<th>1010</th>
<th>1011</th>
<th>1100</th>
<th>1101</th>
<th>1110</th>
<th>1111</th>
</tr>
</thead>
</table>
Properties of ones’ complement

• 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 different representations of zero!
  • E.g., using 4 bits, 1111 and 0000 both represent zero!
Two’s complement

• If integer \( k \) is represented by bits \( b_{n-1}...b_0 \), then \(-k\) is represented by
  \( 100...00 - b_{n-1}...b_0 \) (where \( |100...00| = n+1 \))
  
  • E.g., using \( n = 4 \) bits:
    • \( 6 = 0110 \)
    • \(-6 = 10000-0110 = 1010 = (1111-0110)+1 \)
  
  • Equivalent to taking ones’ complement and adding 1
    • 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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0101</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0110</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Properties of two’s complement

• Same implementation of arithmetic operations as for 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 more negative numbers than positive numbers

• Most common representation of negative integers
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 n bits
  • Invert the bits
  • Add 1
  • E.g., to encode -5 using 8 bits
    • 5 = 00000101 using 8 bits
    • Invert the bits: 11111010
    • Add one: 11111010 + 1 = 11111011
    • -5 encoded in two’s complement using 8 bits is 11111011
Converting to and from two’s complement

• 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
    • Add 1
  • Or
    • subtract 1
    • invert bits
• E.g., 110010
  • Subtract one: 110010 - 1 = 110001
  • Invert the bits: 001110 = 14
  • 110010 encodes -14
• Pick one method and stay with it. I prefer the “flip the bits and add one” way.
Two’s Complement Interpretation

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

<table>
<thead>
<tr>
<th>-2³ = -8</th>
<th>2² = 4</th>
<th>2¹ = 2</th>
<th>2⁰ = 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?
What is going on with this cartoon?

The person in the cartoon is counting sheep using a 16-bit two's complement number!