Binary Number Representations

- Phys4051
- Fall 2003

Introduction

- Why?
  - Storage
  - Arithmetic
  - Interpretations / Decoding

- The (binary) 1001 1001 representation corresponds to which of the following decimal values:
  a) 99
  b) 153
  c) -103

Concepts & Notation

- Byte vs. Bit:
  - 1 Byte = 8 Bits
- Base Notation:
  - 1101\(_{10}\) vs. 1101\(_{2}\)
  - 4051\(_{10}\)
  - What is 0.11\(_{2}\) in \(_{10}\)?

UNSIGNED Binary Number Representations (1)

- Size / Limitations / Advantages
  1. ASCII (Unicode)
    - One Byte (Actually only 7 bits needed)
    - '0' = 48 \(_{10}\), '1' = 49 \(_{10}\), '2' = 50 \(_{10}\) etc.
      (a' = 65 \(_{10}\), 'b' = 66 \(_{10}\) etc.)
    - 49 + 50 = 51
    - 4051\(_{10}\) =

UNSIGNED Binary Number Representations (2)

- BCD (Binary Coded Decimals)
  - Each (decimal) digit is converted to binary
  - Example: 4051\(_{10}\)=
  (BCD)
  - Applications: Displays
  - Needs 4 bits for each digit
  - Mathematically not very convenient

UNSIGNED Binary Number Representations (3)

- (Unsigned) Binary
  - Computer stores ALL numbers ultimately in binary representation anyway
  - Efficient for mathematical operations because each bit position is directly related to its base
4. Hexadecimal (Hex) Base 16
   - '0' = 0, '1' = 1, etc, '9' = 9, '10' = A, '11' = B, '12' = C, '13' = D, '14' = E, '15' = F
   - Notations: 0xABCD, hABCD, ABCDh or $ABCD$
   - Still stored in binary
   - Example: what is 0x10 in base 10?

Conversion Methods (1)
1. Modulus (Remainder) Operation with the (smallest) Base Value
   - Keep dividing by the base value and keep track of the remainders
   - Example: 23|10 = \?|2

Conversion Methods (2)
2. Modulus (Remainder) Operation with the Largest Base Power
   - Keep dividing by the largest base powers
   - Example: 23|10 = \?|2

Conversion Methods (3)
3. Hexadecimal Notation
   - Convenient for large(r) numbers
   - Go from Decimal to Hex to Binary or vice versa
   - Use previous method and remember the hex base powers: 16, 256, 4096 etc.
   - Example: 23|10 = \?|16 = \?|2

Conversion Methods (4)
3. More Hexadecimal Examples
   - Example: 4051|10 = \?|16 = \?|2
   - General Solution:
     - Determine the smallest fractional value, x, that can be expressed in n bits
     - Multiply the original value by 1/x
     - Convert to (hex and then to) binary
     - Add Decimal Point

Conversion Methods (5)
3. Fractional Example:
   - Express 0.3333333|10 in base 2 using a maximum 8 bits (not counting decimal point)
   - General Solution:
     - Determine the smallest fractional value, x, that can be expressed in n bits
     - Multiply the original value by 1/x
     - Convert to (hex and then to) binary
     - Add Decimal Point
SIGNED Binary Number Representations (1)

- If you want to use it in mathematical operations it must obey the following rule:
  \[ Y + (-Y) = 0 \]
- Signed Magnitude Representation
- 2’s Complement

SIGNED Binary Number Representations (2)

- Signed Magnitude (3 Bit Example)
  - First Bit Indicates Sign (0 is +, 1 is -)

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

SIGNED Binary Number Representations (3)

- 2’s Complement (3 Bit Example)
  - Find a system so that \( Y + (-Y) = 0 \)

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

SIGNED Binary Number Representations (4)

- 2’s Complement Formal Approach:
  - Find One’s Complement, i.e., negate each bit separately
  - Add 1 to One’s Complement: This is now Two’s Complement
  - Check: Add two’s complement to the original number: the result should be 0 (except for the overflow bit)

SIGNED Binary Number Representations (5)

- Example find –3 using 2’s Complement (using 3 Bits):
  1: Find +3 Binary
  2: Find its 1’s Complement
  3: Add 1; this is 2’s complement or -3
  4: Check: Add +3 and –3 (check prev. table)

SIGNED Binary Number Representations (6)

- Example find –5 using 2’s Complement (using 3 Bits):
  1: Find +5 Binary
  2: Find its 1’s Complement
  3: Add 1; this is 2’s complement or -5
  4: Check: Add +5 and –5 Conclusion?
**SIGNED Binary Number Representations (7)**

- Example find \(-1\) using 2’s Complement (using 8 Bits):
  1: Find +1 Binary
  2: Find its 1’s Complement
  3: Add 1; **this is 2’s complement or \(-1\)**
  4: Check: Add +1 and \(-1\)

**Binary Arithmetic**

- Addition / Subtraction
- Multiplication
- Left Shift / Right Shift

**Binary Arithmetic (2)**

- Multiplication: \(5 \times 4 = ?\)
  
  101 x 100

**Binary Arithmetic (3)**

- Shift Operations
  - Example 1: Left shift \(3_{10}\)
  - Example 2: Right shift \(13_{10}\)

**Floating Point Representation**

- Example: 4 Byte Floating Point Number in C has a range of +/- 3.4E+/-38 with 7 bits accuracy in the mantissa. How is it implemented?
  
  \(10\)_{10} = 0x 41 20 00 00 = 0100 0001 0010 0000 0000 0000 0000 0000
  
  ^ EXPON. ^ MANTISSA

**Conclusions**

- Computer stores only binary values; its interpretation (or decoding) is up to us
- ASCII (Unicode) vs. Binary Representation
- Signed vs. Unsigned Number Representation
- Mathematical operations with binary numbers, for example, 2’s Complement