Logistics

- Always read the Calendar at http://www.cs.washington.edu/370
- HW1 is posted on the web in the calendar --- due 1/14 11:30am
- Email list: please sign up on the web via the link from the course homepage
- TA Office Hours:
  - Josh Snyder Mondays 3:30-4:20, CSE 220
  - Aaron Miller Tuesdays 12:30-1:20, CSE 220
- My Office Hours:
  - Mondays 12:20-1:00, CSE 668 (grab me after class)
  - TBA

Lecture 2: Number Systems

- Last lecture
  - Class introduction and overview
- Today
  - Binary numbers
  - Base conversion
  - Number systems
  - Twos-complement
  - A/D and D/A conversion

There are 10 kinds of people in the world. Those who understand binary... and those who don’t.

The “WHY” slide

- Binary numbers
  - All computers work with 0's and 1's so it is like learning alphabets before learning English
- Base conversion
  - For convenience, people use other bases (like decimal, hexadecimal) and we need to know how to convert from one to another.
- Number systems
  - There are more than one way to express a number in binary. So 1010 could be 10, -2, -5 or -6 and need to know which one.
- A/D and D/A conversion
  - Real world signals come in continuous/analog format and it is good to know generally how they become 0's and 1's (and vice versa).

Digital

- Digital = discrete
  - Binary codes (example: BCD)
  - Decimal digits 0-9
- Binary codes
  - Represent symbols using binary digits (bits)
- Digital computers:
  - I/O is digital
  - ASCII, decimal, etc.
  - Internal representation is binary
  - Process information in bits

Decimal

<table>
<thead>
<tr>
<th>Decimal</th>
<th>BCD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
</tr>
</tbody>
</table>

The basics: Binary numbers

- Bases we will use
  - Binary: Base 2
  - Octal: Base 8
  - Decimal: Base 10
  - Hexadecimal: Base 16 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- Positional number system
  - 1011 = 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0
  - 63_8 = 6x8^1 + 3x8^0
  - A1_{16} = 10x16^1 + 1x16^0
- Addition and subtraction
  - 1011
    +1010
    ————
    0010
  - 1010
    +0101
    ————
    1011

Binary → hex/decimal/octal conversion

- Conversion from binary to octal/hex
  - Binary: 10011110001
  - Octal: 10 011 110 001 = 2361_{10}
  - Hex: 100 1111 0001 = 4F1_{16}
- Conversion from binary to decimal
  - 1011_2 = 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = 5_{10}
  - 63_{10} = 6x8^1 + 3x8^0 = 51.5_{10}
  - A1_{16} = 10x16^1 + 1x16^0 = 161_{10}
Decimal → binary/octal/hex conversion

<table>
<thead>
<tr>
<th>Binary</th>
<th>Octal</th>
<th>Quotient</th>
<th>Remainder</th>
</tr>
</thead>
<tbody>
<tr>
<td>56+2=28</td>
<td>0</td>
<td>56+7=7</td>
<td>0</td>
</tr>
<tr>
<td>28+2=14</td>
<td>0</td>
<td>7+8=8</td>
<td>0</td>
</tr>
<tr>
<td>14+2=3</td>
<td>1</td>
<td>3+1=4</td>
<td>1</td>
</tr>
<tr>
<td>3+2=1</td>
<td>1</td>
<td>56=111000</td>
<td>0</td>
</tr>
<tr>
<td>1+2=0</td>
<td>0</td>
<td>0+70=70</td>
<td>0</td>
</tr>
</tbody>
</table>

Why does this work?
- N=56, Q=56/2=111000/2=11100 remainder 0
- Each successive divide liberates an LSB (least significant bit)

Number systems

- How do we write negative binary numbers?
- Historically: 3 approaches
  - Sign-and-magnitude
  - Ones-complement
  - Twos-complement
- For all 3, the most-significant bit (MSB) is the sign digit
  - 0 = positive
  - 1 = negative
- Twos-complement is the important one
  - Simplifies arithmetic
  - Used almost universally

Sign-and-magnitude

- The most-significant bit (MSB) is the sign digit
  - 0 = positive
  - 1 = negative
- The remaining bits are the number’s magnitude
- Problem 1: Two representations for zero
  - 0 = 0000 and also –0 = 1000
- Problem 2: Arithmetic is cumbersome

Ones-complement

- Negative number: Bitwise complement positive number
  - 0111 ≡ 7
  - 1000 ≡ –7
- Solves the arithmetic problem
- Remaining problem: Two representations for zero
  - 0 = 0000 and also –0 = 1111

Twos-complement

- Negative number: Bitwise complement plus one
  - 0111 ≡ 7
  - 1001 ≡ –7
- Number wheel
- Only one zero!
- MSB is the sign digit
  - 0 = positive
  - 1 = negative
- Adding representations of x and –y
  - Adding representations of x and –y where x, y are positive we get (2^i - 1) + x – y
    - If x < y then x - y < 0 there is a carry and get –ve number
      - Just add the representations if no carry
    - If x > y then x - y > 0 there is a carry and get +ve number
      - Need to add 1 and ignore the 2^i i.e. “add the carry”
    - If x = y then answer should be 0, get 2^i - 1 = 1111111112

Why ones-complement works

- The ones-complement of an 8-bit positive y is 11111111 - y
- What is 11111111?
  - 1 less than 10000000, 2^8 = 256
  - So in ones-complement –y is represented by (2^i - 1) – y

Add Subtratc

<table>
<thead>
<tr>
<th>Add</th>
<th>Subtract</th>
<th>Compare and subtract</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0000</td>
<td>4</td>
</tr>
<tr>
<td>+ 3</td>
<td>+ 0011</td>
<td>– 3</td>
</tr>
<tr>
<td>= 7</td>
<td>= 0111</td>
<td>– 1</td>
</tr>
</tbody>
</table>
Complementing a complement \( \Rightarrow \) the original number

Arithmetic is easy
- Subtraction = negation and addition
  - Easy to implement in hardware

Adding 1 to get the two's-complement represents \(-y\) by \(2^b - y\)
- If there is a carry then that means \(x \geq y\) and dropping the carry yields \(x - y\)
- If there is no carry then \(x < y\) and then we can think of it as \(2^b - (y-x)\)

Answers only correct mod \(2^b\)
- Summing two positive numbers can give a negative result
- Summing two negative numbers can give a positive result

Make sure to have enough bits to handle overflow

Digital systems need to
- Measure analog quantities
- Control analog systems
  - Drive motors, etc
- How do we connect the analog and digital domains?
  - Analog-to-digital converter (A/D)
  - Example: CD recording
  - Digital-to-analog converter (D/A)
  - Example: CD playback
Sampling

- **Quantization**
  - Conversion from analog to discrete values
- **Quantizing a signal**
  - We sample it

Conversion

- **Encoding**
  - Assigning a digital word to each discrete value
- **Encoding a quantized signal**
  - Encode the samples
  - Typically Gray or binary codes