Lecture 2: Binary Number Systems

Digital

Digital = Discrete

- Decimal digits
- DNA nucleotides (CATG)
- Binary codes
- Symbols mapped to bits

Digital Computers

- I/O is digital
- ASCII, decimal, binary, etc.
- Internal representation
- Binary

Number Systems

Bases In This Class
- Binary (2), Octal (8), Decimal (10), Hexadecimal (16)
- Positional numbering systems (“significant digits”)

Adding, Subtracting

“There are 10 kinds of people in the world—those who understand binary numbers, and those who don’t.”

Administrivia

Make sure
- Signed up to the mailing list

Homework
- Will be assigned on Friday prior to due date (so that it can haunt you over the weekend!)
- Homework guru: Adrienne Wang (axwang@cs)
  Office hours: W 3-5pm in CSE 218

Office hours
- Benjamin Ylvisaker (ben8@cs)
  Office hours: Th 1:30-3:30 in CSE 003
### Conversions

**Binary to Octal and Hexadecimal**

\[ 1011011001_2 = \overset{2}{\overline{1011}} \overset{2}{\overline{0110}} \overset{2}{\overline{001}} = 133_8 \]

\[ 1011011001_2 = \overset{8}{\overline{1011}} \overset{8}{\overline{0110}} \overset{8}{\overline{001}} = 2D9_{16} \]

**Octal and Hexadecimal to Binary**

\[ 401_8 = \overset{2}{\overline{10}} \overset{2}{\overline{0}} \overset{2}{\overline{0}} \overset{2}{\overline{1}} = 10000001_2 \]

\[ B10_{16} = \overset{4}{\overline{1111}} \overset{4}{\overline{0001}} = \overset{2}{\overline{11110001}}_2 \]

### Decimal to Others

**Decimal to Binary**

\[ \frac{58}{2} = 29 \text{ remainder } 0 \]
\[ \frac{29}{2} = 14 \text{ remainder } 1 \]
\[ \frac{14}{2} = 7 \text{ remainder } 0 \]
\[ \frac{7}{2} = 3 \text{ remainder } 1 \]
\[ \frac{3}{2} = 1 \text{ remainder } 1 \]
\[ \frac{1}{2} = 0 \text{ remainder } 1 \]

\[ 111010_2 = 11101_2 \]

Why does this work?

\[ \frac{111010}{2_{10}} = 11101 \text{ remainder } 0 \]
\[ \frac{11101}{2_{10}} = 111 \text{ remainder } 1 \]

**Decimal to Octal**

\[ \frac{58}{8} = 7 \text{ remainder } 2 \]
\[ \frac{7}{8} = 0 \text{ remainder } 7 \]

### Negative Numbers

- Negative binary numbers?

- Historically
  - sign/magnitude
  - ones-complement
  - twos-complement

- For all three:
  - most significant bit (msb) is the sign
    - 0=positive, 1=negative
  - twos-complement universally most used
    - simplifies arithmetic

### Sign/Magnitude

- Most significant bit is sign
  - 0=positive, 1=negative
  - remaining bits are magnitude

- For all three:
  - Problem 1: two zeros!
    - 0000_2 = 010 and 1000_2 = -010 = 010
  - Problem 2: arithmetic is messy (hard to implement)

\[ \begin{align*}
4_{10} &= 0100_2 \\
+3_{10} &= 0011_2 \\
\downarrow_{10} &= 0110_2 = +7_{10} \\
-4_{10} &= 0001_2 \\
-3_{10} &= 1011_2 \\
\downarrow_{10} &= 1110_2 = -7_{10}
\end{align*} \]
Ones-Complement
- Most significant bit is sign
  - 0 = positive, 1 = negative
- Negative number is positive numbers bitwise complement

\[
3_{10} = \overline{11}_{2} \\
-3_{10} = \overline{100}_{2}
\]

Problem 2: Arithmetic is clean (add carry)

\[
\begin{align*}
4_{10} &= 0100_{2} \\
+3_{10} &= 0011_{2} \\
\hline
7_{10} &= 1111_{2}
\end{align*}
\]

\[
\begin{align*}
-4_{10} &= 1100_{2} \\
+3_{10} &= 0011_{2} \\
\hline
-1_{10} &= 1111_{2}
\end{align*}
\]

Problem 1: Still two zeros!

\[0000_{2} = 0_{10} \text{ and } 1111_{2} = -0_{10} = 0_{10}\]

Twos-Complement
- Most significant bit is sign
  - 0 = positive, 1 = negative
- Negative number is bitwise complement plus 1

\[
\begin{align*}
3_{10} &= \overline{11}_{2} + 1 \\
-3_{10} &= \overline{100}_{2} + 010 = 010_{2}
\end{align*}
\]

Twos-Complement Math
- Arithmetic works (drop carry)

\[
\begin{align*}
4_{10} &= 0100_{2} \\
+3_{10} &= 0011_{2} \\
\hline
7_{10} &= 1111_{2}
\end{align*}
\]

\[
\begin{align*}
-4_{10} &= 1100_{2} \\
+3_{10} &= 0011_{2} \\
\hline
-1_{10} &= 1111_{2}
\end{align*}
\]

Twos-Complement Exercise
- Test your skills convert 110 and \(-5\) to 4 bit twos-complement binary and then add them

\[1_{10} = \boxed{-5_{10}} = \boxed{0111}_{2}\]
Twos-Complement Overflow

- Numbers may add out of range (overflow)

```
0000
0001
0011
1111
1110
1100
1011
1010
1000 0111
0110
0100
0010
0101
0110
1001
1101
-8
-7
-6
-5
-4
-3
-2
-1

+4=0100
+6=0110
+10=1010
```

Twos-Complement Overflow

- Numbers may add out of range (overflow)

```
carry bits
0100
+4=0100
+6=0110
+10=1010
11000
+4=0100
+3=1110
+1=1000
```

Last two carry bits: $c_{\text{last}}$ and $c_{2\text{last}}$

<table>
<thead>
<tr>
<th>$c_{\text{last}}$</th>
<th>$c_{2\text{last}}$</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Twos-Complement Misc

- Sign extension
  - $+6_{10} = 0110_2$
  - $-6_{10} = 1001_2$
- Extend to eight bits (a byte):
  - $+6_{10} = 00000110_2$
  - $-6_{10} = 11111001_2$
- Different binary numbers have different values
  - $11001 = 2S_{10}$ unsigned
  - $11001 = -9_{10}$ sign/magnitude
  - $11001 = -6_{10}$ ones-complement
  - $11001 = -7_{10}$ twos-complement
- The weird number: $1110_2 = -8$

Machine Independent?

- HAKMEM Item 154 (Bill Gosper)

The myth that any given programming language is machine independent is easily exploded by computing the sum of powers of 2.

If the result loops with period = 1 with sign +, you are on a sign-magnitude machine.
If the result loops with period = 1 at -1, you are on a twos-complement machine.
If the result loops with period > 1, including the beginning, you are on a ones-complement machine.
If the result loops with period > 1, not including the beginning, your machine isn't binary -- the pattern should tell you the base.
If you run out of memory, you are on a string or Bignum system.
If arithmetic overflow is a fatal error, some fascist pig with a read-only mind is trying to enforce machine independence. But the very ability to trap overflow is machine dependent.
**Switches**

- Implementing a simple circuit (arrow shows action if wire changes to “1”):
  
  - close switch (if A is “1” or asserted) and turn on light bulb (Z)
  
  - open switch (if A is “0” or unasserted) and turn off light bulb (Z)

  \[ Z = A \]

**Switching Networks**

- Switch settings determine whether a conducting network to a light bulb
- Larger computations?
  
  - Use a light bulb (output) to set other switches (input)
  - Example: Mechanical relay

**Transistor Networks**

- Relays no more: slow and big
- Modern digital electronics predominately uses CMOS technology
  
  - MOS: metal-oxide semiconductor
  - C: complementary (both p and n type transistors arranged so that power is dissipated during switching.)
MOS Transistors

- MOS transistors have three terminals: drain, gate, and source
- Act as switches: if the voltage on the gate terminal is (some amount) higher/lower than the source terminal then a conducting path will be established between the drain and source terminals.

MOS Transistors

n-channel
- open when voltage at G is low
- closes when: \( \text{voltage}(G) > \text{voltage}(S) + \varepsilon \)

p-channel
- closed when voltage at G is low
- opens when: \( \text{voltage}(G) < \text{voltage}(S) - \varepsilon \)

MOS Networks

Two Input Networks

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y )</th>
<th>( z_1 )</th>
<th>( z_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 volts</td>
<td>0 volts</td>
<td>0 volts</td>
<td>0 volts</td>
</tr>
<tr>
<td>0 volts</td>
<td>3 volts</td>
<td>3 volts</td>
<td>3 volts</td>
</tr>
<tr>
<td>3 volts</td>
<td>0 volts</td>
<td>3 volts</td>
<td>3 volts</td>
</tr>
</tbody>
</table>

What is the relationship between \( x \), \( y \), and \( z \)?

Your To Do List

- Things Internet
  - Sign up for mailing list

- Things Reading
  - Week 1 reading (on website): pp.1-27, Appendix A, pp.33-46

- Things Homework
  - Homework 1 posted on website (due this Friday)

- Things Laboratory
  - Attend first lab session if you haven’t already