Number Representation

How do we represent data in a computer?

- At the lowest level, a computer is an electronic machine.
  - works by controlling the flow of electrons

- Easy to recognize two conditions:
  1. presence of a voltage – we’ll call this state “1”
  2. absence of a voltage – we’ll call this state “0”

- Could base state on *value* of voltage, but control and detection circuits more complex.
  - compare turning on a light switch to measuring or regulating voltage

Computer is a binary digital system.

- Basic unit of information is the *binary digit*, or *bit*.

What kinds of data do we need to represent?

- **Numbers** – signed, unsigned, integers, floating point, complex, rational, irrational, …
- **Text** – characters, strings, …
- **Images** – pixels, colors, shapes, …
- **Sound**
- **Logical** – true, false
- **Instructions**
- …
Integer Numbers

- Integers are almost universally stored as binary numbers.
- When you enter an integer from the keyboard, software converts the ASCII or Unicode characters to a binary integer.

Integers

- Weighted positional notation
  - like decimal numbers: “329”
  - “3” is worth 300, because of its position, while “9” is only worth 9

Binary Fractions

- Each position is twice the value of the position to the right.

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

1 0 1 0 . 0 0 1

What is this number in decimal?

Binary Arithmetic

- Base-2 addition – just like base-10
  - add from right to left, propagating carry
**Negative Integers**

- Almost all systems for storing negative binary numbers set the left most bit (MSB) to indicate the sign of a number.

  **Common formats:**
  - Signed Magnitude
  - Ones Complement
  - Twos Complement

**Signed Magnitude**

- Negative number are the same as positive with the sign bit set.

  **Three bit example**

<table>
<thead>
<tr>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>-0</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
</tr>
</tbody>
</table>

- There are two zeroes, positive and negative.

**Signed Magnitude Arithmetic**

- Normal addition does not work for negative signed magnitude numbers.

  \[
  \begin{align*}
  1001 & \quad -1 \\
  + 0100 & \quad +4 \\
  1101 & \quad (-5) \quad 3
  \end{align*}
  \]

**Ones Complement**

- Negative number are the logical inverse of positive numbers.

  **Three bit example**

<table>
<thead>
<tr>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>-3</td>
<td>-2</td>
<td>-1</td>
<td>-0</td>
</tr>
</tbody>
</table>

  Mathematically positive and negative zero are the same, but they are different bit patterns.
Ones Complement Arithmetic

- The carry out of the sign position needs to be added to the number

```
\[
\begin{array}{c|c}
\text{carry} & -2 \\
01 & + \text{110} \\
\hline
+1 & \text{011} \\
\hline
100 & -3
\end{array}
\]
```

Twos Complement

- Negative number are the logical inverse of positive numbers plus 1.

Three bit example

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

Normal binary arithmetic works for positive and negative numbers.

Two’s Complement Representation

- If number is positive or zero,
  - normal binary representation, zero in upper bit
- If number is negative,
  - start with positive number
  - flip every bit (i.e., take the one’s complement)
  - then add one

```
\[
\begin{array}{c}
\text{00101} \ (5) \\
00101 \ (1\text{'s comp}) \\
\hline
+1 \\
\hline
\text{11011} \ (-5)
\end{array}
\]
```

Two’s Complement Shortcut

- To take the two’s complement of a number:
  - copy bits from right to left up to and including the first “1” bit
  - flip remaining bits to the left

```
\[
\begin{array}{c}
\text{01101000} \\
10010111 \ (1\text{'s comp}) \\
\hline
+1 \\
\hline
10011000 \ (-10)
\end{array}
\]
```

```
\[
\begin{array}{c}
\text{10101000} \\
01010111 \ (1\text{'s comp}) \\
\hline
\text{flip} \\
\hline
10011000 \ (copy)
\end{array}
\]
```
Two’s Complement Signed Integers

<table>
<thead>
<tr>
<th>$2^3$</th>
<th>$2^2$</th>
<th>$2^1$</th>
<th>$2^0$</th>
<th>$2^3$</th>
<th>$2^2$</th>
<th>$2^1$</th>
<th>$2^0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

What is the decimal value of the 6-bit two’s complement number?

110011

A. -3  
B. 33  
C. -11  
D. -12  
E. -13

2’s Complement Addition

- Use normal binary addition regardless of sign.
- Ignore carry out

\[
\begin{align*}
01101000 \ (104) &+ 1110000 \ (-16) &\rightarrow 01011000 \ (98) \\
11110110 \ (-10) &+ 1110111 \ (-9) &\rightarrow 11101101 \ (-19)
\end{align*}
\]

Subtraction

- Negate subtrahend (2nd number) and add.

\[
\begin{align*}
00001000 \ &+ 11110000 \ (-16) &\rightarrow 11110110 \ (-10) \\
-0000101 \ (-5) &+ 11110111 \ (-9) &\rightarrow 00000011 \ (3)
\end{align*}
\]

Assuming 8-bit 2’s complement numbers.
### Sign Extension

- When moving an integer into a larger register the upper bits must be set to the sign bit.
- If we just pad with zeroes on the left:
  - 4-bit 8-bit
    - 0100 (4) 00000100 (still 4)
    - 1100 (-4) 00001100 (12, not -4)
- Instead, replicate the MS bit -- the sign bit:
  - 4-bit 8-bit
    - 0100 (4) 00000100 (still 4)
    - 1100 (-4) 11111100 (still -4)

### Overflow

- If operands are too big, then sum cannot be represented as an \( n \)-bit 2’s complement number.

\[
\begin{array}{c|c|c}
\text{carry into sign bit} & \text{carry out of sign bit} \\
01000 & 11000 \\
01001 & 10111 \\
10001 & 01111 \\
\end{array}
\]

- We have overflow if:
  - signs of both operands are the same, and
  - sign of sum is different.
- Another test -- easy for hardware:
  - carry into left most bit does not equal carry out

### Decimal Numbers

- Some systems (i.e. Intel Pentium) support decimal numbers in packed or unpacked format.

**packed decimal** uses 4 bit fields for each digit

\[
9375 = 1001,0011,0111,0101
\]

**unpacked decimal** uses a byte per digit (ASCII)

\[
9375 = 00111001,00110011,00110111,00110101
\]

### What is printed by this program?

```java
int num;
num = 100000000;
while (num > 0) {
    num = num + 100000000;
}
System.out.println(num);
```

1. Runs forever
2. zero
3. large negative number
4. -1
### Number Order

<table>
<thead>
<tr>
<th></th>
<th>bits changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>011</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>2</td>
</tr>
<tr>
<td>111</td>
<td>1</td>
</tr>
</tbody>
</table>

- With normal binary numbers, many bits may change from number to number as you count up.

### Gray Code

<table>
<thead>
<tr>
<th></th>
<th>bits changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>011</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
</tr>
<tr>
<td>111</td>
<td>1</td>
</tr>
</tbody>
</table>

- With gray code numbers, only 1 bit changes from number to number as you count up.

### Gray Code Input

- When reading sensors to measure the position of something, gray codes reduce the probability of incorrect input.

### Counting in Gray Code

- To make the list twice as long, copy and reverse the list and append a 1 bit to the left.