Numbers, Logic, and Binary

Gabriel Hugh Elkaim
Positional Number Systems

1976_{10} \text{ in base 10}

10^3 \ 10^2 \ 10^1 \ 10^0

\begin{align*}
\text{octal (base 8)} &: 723_8 \\
8^2 &+ 2 \times 8^1 &+ 3 \times 8^0 &= 69 + 16 + 3 = 88_{10}
\end{align*}

\begin{align*}
2 \times 9 + 1 \times 3 + 0 \times 1 &= 21_{10} \\
&+ 35 \\
&= 105_{10}
\end{align*}
Any base "b" 
\[ b^2 b' b^0 \]
\[ 10|0|0|0|0| \]
\[ \sum_{p=0}^{n} d_p b^p \]
\[ _{p=0} \]
Arabic/Indic Numerals

Base (Radix) = 10

Alphabet/digits/symbols = 0...9

Introduced to Europe by Leonardo Fibonacci ~ 1202
Arabic/Indic Numerals

- The Italian mathematician Leonardo Fibonacci
- Also known for the Fibonacci sequence
  - 1, 1, 2, 3, 5, 8, 13, 21

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>European</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Arabic-Indic</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Eastern Arabic-Indic</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Persian and Urdu)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Devanagari</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Hindi)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tamil</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Image of Leonardo Fibonacci]
Computers Do Many Things

A powerful calculator! Made of switches!

It follows instructions!

An input output device!

Exotic hardware!
Cooking Follows a Recipe

- Ingredients or Input
- A Processing Unit Under Control
- Output

Ugh! What is this?!?
Looking at it a Different Way

Or, more abstractly:

**CONTROL**

**INPUT** → **PROCESSING UNIT** → **OUTPUT**

White arrows (⇒) are the flow of food.
Gray arrow (→) is the flow of information.
Black arrow (←) is the flow of control.
Computers Follow Recipe, Too

Von Neumann’s Idea:

Reads Program
Decision Sequence of Steps

Control ↔ Processing Unit

Input → Memory → Output

Let’s See you

Run Data Instructions

Stores Inputs Results From Processing

Information Flow

Control Flow
Elementary Operations are Logical

The computer's elementary operations are LOGICAL.

1. You are a man.
2. All men are mortal.
3. Therefore, you are mortal.

Ahem? How do you know all men are mortal??
Symbolic Logic

This step was taken by George Boole (1815-1864), an English mathematician who built an "algebra" out of logic.

And, or, not!
Statements are True or False

\[ P = \text{"The pig has spots."} \]

\[ Q = \text{"The pig is glad."} \]
Compound Statements

Now form the compound sentences:

- \( P \land Q \): The pig is spotted \textbf{AND} the pig is glad.
- \( P \lor Q \): The pig is spotted \textbf{OR} the pig is glad.

When are these sentences true?
All Possible Variations

P TRUE, Q TRUE

P FALSE, Q TRUE

P TRUE, Q FALSE

P FALSE, Q FALSE
<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

**More Formalism**

<table>
<thead>
<tr>
<th>T</th>
<th>T</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>P T</td>
<td>Q T</td>
<td>P F</td>
</tr>
</tbody>
</table>

- **P TRUE, Q TRUE**
- **P FALSE, Q TRUE**
- **P TRUE, Q FALSE**
- **P FALSE, Q FALSE**
AND

"The pig is glad AND has spots."

This is true only in the one case in which P, Q are both true. This is summarized in a truth table:

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P AND Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
"The pig is glad OR has spots."

This is true in the three cases for which either one of the statements $P$, $Q$ is true.

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$P \lor Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
</tr>
</tbody>
</table>
NOT

$\neg p = \text{The pig is NOT spotted.}$

This operator simply turns a statement into its opposite.

<table>
<thead>
<tr>
<th>$p$</th>
<th>$\neg p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>P</td>
<td>Q</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P OR Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P</th>
<th>NOT P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ T = 1 \]
\[ F = \emptyset \]
How Do Computers Compute?

NAMELY: HOW DO COMPUTERS COMPUTE?

AS SURE AS I DIDN'T STEAL CALCULUS FROM NEWTON!
Numbers for Computers

- Many ways to represent a number
- Representation does not affect computation result
- Representation does affect how difficult it is to compute results.
- Computer needs a representation that works with (fast) electronic circuits
- Computers generally only have 2 states.
Binary Number System

The system is called BINARY NUMBERS.

I'd count by fours, but I only have one free paw!

Tree sloths always count in binary!

They're based on two!

Gabriel Hugh Elkaim
Binary Number System

Base (radix): 2

Digits/Symbols: 0, 1 — “Bit”

Example:

\[ \begin{align*}
2^3 & \quad 2^2 & \quad 2^1 & \quad 2^0 \\
8 & \quad 4 & \quad 2 & \quad 1 \\
1011_2 & \quad & \quad & \quad \\
1011_2 & = & 8 + 2 + 1 = 11_{10}.
\end{align*} \]

\[ \begin{align*}
2^4 & \quad 2^3 & \quad 2^2 & \quad 2^1 & \quad 2^0 \\
16 & \quad 8 & \quad 4 & \quad 2 & \quad 1 \\
100100_2 & = & 8 + 16 = 24_{10}.
\end{align*} \]
IN ACTUALITY, "10" MEANS:

1 (ONE) HANDFUL* AND
0 (ZERO) FINGERS LEFT OVER

10 BINARY = 2 DECIMAL

TWO TWO TWO

TWO TWO TWO

TWO TWO TWO

TWO TWO TWO

*Must be right hand and there must be more than one finger.
Knowing The Powers Of Two

\[ 1 = 2^0 = 1 \]
\[ 10 = 2^1 = 2 \]
\[ 100 = 2^2 = 4 \]
\[ 1000 = 2^3 = 8 \]
\[ 10000 = 2^4 = 16 \]
\[ 100000 = 2^5 = 32 \]
\[ 1000000 = 2^6 = 64 \]
\[ 10000000 = 2^7 = 128 \]
\[ 100000000 = 2^8 = 256 \]
\[ 1000000000 = 2^9 = 512 \]
\[ 10000000000 = 2^{10} = 1024 \]
Converting Binary to Decimal

\[
\begin{array}{c|c}
\text{IN DECIMAL:} & \text{IN BINARY:} \\
497 & 111110001 \\
400 & \overline{256} \\
+90 & 128 \\
+7 & 64 \\
\hline
174 & 32 \\
10 & 16 \\
\hline
\end{array}
\]

10101110

128 64 32 16 8 4 2

\[
\begin{array}{c}
1 \\
2 \\
\hline
174_{10}
\end{array}
\]
Another Way of Doing It

\[ \begin{array}{cccccccc}
... & 2^9 & 2^8 & 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 \\
\hline
1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0
\end{array} \]

\[ 256 + 16 + 8 + 2 = 282 \]
Counting in Binary

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>111</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
</tr>
<tr>
<td>1100</td>
<td>12</td>
</tr>
<tr>
<td>1101</td>
<td>13</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
</tr>
<tr>
<td>1111</td>
<td>15</td>
</tr>
<tr>
<td>10000</td>
<td>16</td>
</tr>
<tr>
<td>10001</td>
<td>17</td>
</tr>
<tr>
<td>10010</td>
<td>18</td>
</tr>
<tr>
<td>10011</td>
<td>19</td>
</tr>
<tr>
<td>10100</td>
<td>20</td>
</tr>
<tr>
<td>...</td>
<td>Etc!</td>
</tr>
</tbody>
</table>

Easy as falling out of bed!
Octal Number System

Base (Radix) = 8

Digits/Symbols = 0, 1, ..., 7

8^0 8^1 8^2 8^3 8^4 8^5 8^6

345_8 = 3 \times 64 + 4 \times 8 + 5 \times 1 = 229_{10}

1001_8 = 1 \times 512 + 0 + 0 + 1 = 513_{10}

In C, octal number has a leading 0. 01001
Hexadecimal Number System

HEXADECIMAL, or BASE-16, NUMERALS.

10_{hex} = 16_{decimal}
100_{hex} = 16^2 = 256
1000_{hex} = 16^3 = 4096

Etc!

<table>
<thead>
<tr>
<th>DECIMAL</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEX</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
</tbody>
</table>

Our Favorite!
Hexadecimal (HEX)

Base/index = 16

Symbols/digits = 0-9, A-F

Example:

\[ 3F_{16} = 3 \times 16 + 15 = C3_{10} \]

\[ 2E8_{16} = 3 \times 256 + 14 \times 16 + 8 \times 1 = 1000_{10} \]

In C, hex is indicated by a leading 0x.
Examples of Converting Hex to Decimal

\[ 0 \times 321 = 3 \times 256 + 2 \times 16 + 1 \]
\[ 768 + 32 + 1 = 801_{10} \]

\[ 0 \times \text{CAFE} = 12 \times 9096 + 10 \times 256 + 15 \times 16 \]
\[ + 14 \]
\[ 49152 + 2560 + 240 + 14 \]
\[ 51960_{10} \]
Announcements

1. Waitlist is over.
2. Labs 3-6 grading done by today.
3. Website for studying MAP for UB 1 is finished.
Base Conversion

1. From any base to base 10.
2. From base 10 to any other base $B$.
3. From one base $B$ to another base $C$. 
From Base $b$ to Base 10

Base $(radix)$: $b$

Digits (numens): $0 \rightarrow (b-1)$

... $d_2 \ d_1 \ d_0$

Value: $\sum_{i=0}^{n} d_i \times b^i$

\[
\begin{align*}
123_{b} & = 125 + 2 \times 25 + 3 \times 5 + 9 = 194_{10}.
\end{align*}
\]
From Base $b$ to Base 10

- Use successive division
- Remember the remainders
- Divide again with the quotient

$31020_5 = 2010_{10}$
\[ \frac{1967}{4} = 491 \text{ R } 3 \]

\[ \frac{491}{9} = 122 \text{ R } 3 \]

\[ \frac{122}{4} = 30 \text{ R } 2 \]

\[ \frac{30}{9} = 3 \text{ R } 2 \]

\[ \frac{7}{9} = 1 \text{ R } 3 \]

\[ \frac{3}{4} = 0 \text{ R } 1 \]
From Base 10 to Base $b$ – Method 1

194_{10} \rightarrow \_5

\begin{align*}
i & = 0 \quad N = 194 \quad b = 5 \\
q & = 38 \quad r = 4
\end{align*}

\begin{align*}
i & = 1 \quad N = 38 \quad b = 5 \\
q & = 7 \quad r = 3
\end{align*}

\begin{align*}
i & = 2 \quad N = 7 \quad b = 5 \\
q & = 1 \quad r = 2
\end{align*}

\begin{align*}
i & = 3 \quad N = 1 \quad b = 5 \\
q & = 0 \quad r = 1
\end{align*}

r = \text{ith digit of } N_b

q = N/b

++i

N = q

\begin{align*}
q & = 0 \\
\text{END}
\end{align*}

1234_5
From Base 10 to Base $b$ – Method 1

1. $i = 0$
2. $q = \frac{N}{b}$, $r = N \% b$
3. $++i$
4. $N = q$
5. $r =$ ith digit of $N_b$
6. $q = 0$?
   - if yes, END
   - if no, go back to step 2
From Base 10 to Base $b$ – Method 2

- Know powers of $b$
- Subtract and lay out powers of $b$
- Multiply by scalar digit (1/0)
- Put that scalar digit in position
- Repeat w/ remainder

$312_{10} \rightarrow ?_2$
From Base 10 to Base $b$ – Method 2

$312_{10}$

\[ \begin{array}{c|c|c|c|c|c} 
& 4 & 6 & 3 & 6 & 2 \\
\hline
& 100111000 \\
\end{array} \]

$312 - 256$ $=$ $56$

$56 - 32$ $=$ $24$

$24 - 16$ $=$ $8$

$8 - 2^0$ $=$ $0$

$1 \times 2^8$

$1 \times 2^5$

$1 \times 2^3$

$1 \times 2^2$

$1 \times 2^1$

$1 \times 2^0$

$8$

$2$

$8$

$0$

$1$

$0$

$1$
From Base $b$ to Base $c$

- Use an intermediate base $d$ that is known
- Base $b \rightarrow$ Base $10 \rightarrow$ base $c$.

In some cases it is easier to go through base 2.
Decimal To Binary Conversion: Method 1

\[ 444_{10} \rightarrow ?_2 \]

\[ \begin{array}{c}
\frac{444}{2} = 222 & \text{R} & 0 \\
\frac{222}{2} = 111 & \text{R} & 0 \\
\frac{111}{2} = 55 & \text{R} & 1 \\
\frac{55}{2} = 27 & \text{R} & 1 \\
\frac{27}{2} = 13 & \text{R} & 1 \\
\frac{13}{2} = 6 & \text{R} & 1 \\
\frac{6}{2} = 3 & \text{R} & 0 \\
\frac{3}{2} = 1 & \text{R} & 1 \\
\frac{1}{2} = 0 & \text{R} & 1
\end{array} \]

\[ 110111100_2 \]
Decimal To Binary Conversion: Method 2

128  64  32  16  8  4  2  1

61_{10} = ?_{2}

\[ \frac{-29}{16} = -1 \]
\[ \frac{-5}{5} = -1 \]
\[ \frac{-4}{-4} = 1 \]

111101
Binary to Octal Conversion

- Group in 3 bits at a time

  START AT THE LEAST SIGNIFICANT BIT (LSB)

- Pad w/ leading zeros as required

- Write one octal digit per group.
Binary to Octal Conversion: Examples

\[ 1001011_2 \]
\[ 427_8 \]

\[ 0011010_2 \]
\[ 12_8 \]

\[ 324_8 \]
\[ 011 \quad 010 \quad 100 \]
\[ 11010100_2 \]
Octal to Binary Conversion

6C6₈ → ?₂

110 110 110

110110110₂
**Binary to Hex Conversion**

- Group in 4 bits starting from LSB
- Add leading zeros as required
- Write one hex digit per group:

<table>
<thead>
<tr>
<th>Bin</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>100</td>
<td>6</td>
</tr>
<tr>
<td>101</td>
<td>7</td>
</tr>
<tr>
<td>110</td>
<td>8</td>
</tr>
<tr>
<td>111</td>
<td>9</td>
</tr>
<tr>
<td>1000</td>
<td>A</td>
</tr>
<tr>
<td>1001</td>
<td>B</td>
</tr>
<tr>
<td>1010</td>
<td>C</td>
</tr>
<tr>
<td>1011</td>
<td>D</td>
</tr>
<tr>
<td>1100</td>
<td>E</td>
</tr>
<tr>
<td>1101</td>
<td>F</td>
</tr>
<tr>
<td>1110</td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td></td>
</tr>
</tbody>
</table>
Binary to Hex Conversion: Examples

1111 1110 1101 0100 | 0000 1111 1110 1000

9 E 7 0

0x9E70

1FA3

0x1FA3
Hex to Binary Conversion

0xCAFÉ

1100101011111111

0x812F

10000001000101111
Conversion Table
More Conversions

HEX → OCTAL?

DECIMAL → HEX:

WHY USE HEX & OCTAL?
Positional Addition

\[
\begin{array}{c}
12_{10} \\
17_{10} \\
\hline
29
\end{array}
\quad \quad \quad \quad
\begin{array}{c}
12_{10} \\
19_{10} \\
\hline
31
\end{array}
\quad \quad \quad \quad
\begin{array}{c}
113_{13} \\
42_{13} \\
\hline
210_{13}
\end{array}
\quad \quad \quad \quad
\begin{array}{c}
5_{10} = 10_{5} \\
\hline
\end{array}
\quad \quad \quad \quad
\begin{array}{c}
1 \\
210_{3} \\
\hline
21_{3}
\end{array}
\quad \quad \quad \quad
\begin{array}{c}
1001_{3}
\end{array}
\]
Addition in base 2

\[ \begin{align*}
28_{10} & \rightarrow 100001 \\
29_{10} & \rightarrow 011101 \\
\text{result} & \rightarrow 111110
\end{align*} \]
A Little Bit on Adding

Binary calculation is simple.
There are only five rules to remember:

\[
\begin{align*}
0 + 0 &= 0 \\
0 + 1 &= 1 \\
1 + 0 &= 1 \\
1 + 1 &= 10 \\
\text{AND THE HANDY FIFTH RULE:} \\
1 + 1 + 1 &= 11
\end{align*}
\]

As opposed to 100 sums in decimal: 9+6, 7+5, 9+3, 8+4, 4+6, etc etc etc etc etc!!!
1
101
1011
0110
1111
100000
Largest Number in $n$ digits

\[ \text{base 10} \rightarrow 9 \quad n=1 \quad 99 \quad n=2 \quad b^n-1 \]

8 \rightarrow 7

16 \rightarrow 1F

For any base, $b^n - 1$
Data Representation

- In books...
- On audio and video disks...
- In paintings or drawings...
- On tape...
- In the human memory...
- How to build & deliver H-bombs
- In diagrams, etc!

Gabriel Hugh Elkaim
BIT is an abbreviation of "Binary digit." It refers to a single 0 or 1.

Is it Binary digit or Binary digit?

It's very common to group bits eight at a time, and any string of eight bits is called a Byte. There are $2^8$, or 256, possible bytes, from 00000000 to 11111111.
Data Representation and Arithmetic

Gabriel Hugh Elkaim
Data Representation

- In books...
- On audio and video disks...
- In paintings or drawings...
- On tape...
- In the human memory...
- How to build & deliver H-bombs in diagrams, etc!
Data Representation

**Goal:** Store information (numbers, characters, ...)

- In Binary
Data Representation

Integers, or whole numbers— if they aren’t too large—are encoded in straight binary. For instance, 185 would become 10111001.

Binary coded decimal represents a number in decimal, but with each digit encoded in binary. For instance, 967 would become 1001 0101 0111 0 6 7.

Floating point representation is for large or fractional numbers. For example, 197,000,030.2 would be encoded as the binary equivalent of 197 5, meaning 197 \times 10^5. Floating point representation often involves rounding off.
BIT is an abbreviation of "Binary digit." It refers to a single 0 or 1.

Is it Binary digit or Binary digit?

It's very common to group bits eight at a time, and any string of eight bits is called a BYTE. There are $2^8$, or 256, possible bytes, from 00000000 to 11111111.
Storing Information (n-bits)

- **25 bits**
  - 11
  - 10
  - 01
  - 00
  - 4 items

- **3 bits**
  - 111
  - 110
  - 101
  - 000
  - 8 items

- **8 bits** → 2^8
  - 256

- **16 bits** → 2^16
  - 65,536
  - 65K

- **32 bits** → 2^32
  - 4,294,967,296
  - 4G

- **64 bits** → 2^64
  - 18,446,744,073,709,551,616
  - 20PB

More bits: 1029
Integer Representation (1.2)

- Assume I have a fixed number of bits (32)
- Which 4 billion integers do I want?

- UNSIGNED $\rightarrow$ 0 $\rightarrow$ 4 Billion consecutive positive integers

- SIGNED $\rightarrow$ 0 $\rightarrow$ computation $\frac{1}{2}$ the range

- 2's complement
  - $\frac{1}{2}$ range $\rightarrow 0$
Integer Representation (2.2)

Most Negative

Most Positive Numbers
Unsigned Integers (2.2)
One’s Complement
One’s Complement Representation
Two’s Complement (1.2)
Two’s Complement (2.2)
Two’s Complement Conversion
Two’s Complement Addition (1.2)
Two’s Complement Addition (2.2)
Subtraction (1.2)
Subtraction (2.2)
Two’s Complement Subtraction (1.2)
Two’s Complement Subtraction (2.2)
Sign Extension (1.2)
Sign Extension (2.2)
Sign Extension — Unsigned
Sign Extension — 1’s and 2’s Complement