Number Representation and Operators
THERE ARE 10 TYPES OF PEOPLE IN THE WORLD, THOSE WHO UNDERSTAND BINARY AND THOSE WHO DON'T......
Decimal Numbering System

Ten symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Represent larger numbers as a sequence of digits
  • Each digit is one of the available symbols

Example: 7061 in decimal (base 10)
  • $7061_{10} = (7 \times 10^3) + (0 \times 10^2) + (6 \times 10^1) + (1 \times 10^0)$
Octal Numbering System

Eight symbols: 0, 1, 2, 3, 4, 5, 6, 7

• Notice that we no longer use 8 or 9

Base Comparison:

Base 10: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12...

Base 8: 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14...

Example: What is 15₈ in base 10?

• $15₈ = (1 \times 8^1) + (5 \times 8^0) = 13_{10}$

Example: What is 7061₈ in base 10?

• $7061₈ = (7 \times 8^3) + (0 \times 8^2) + (6 \times 8^1) + (1 \times 8^0) = 3633_{10}
Binary Numbering System

Two symbols: 0, 1

Convention: $2_{10} = 10_2 = 0b10$

Example: What is $0b110$ in base 10?
• $0b110 = 110_2 = (1 \times 2^2) + (1 \times 2^1) + (0 \times 1^0) = 6_{10}$
Hexadecimal Number System

Hexadecimal is base 16 (>10)

Sixteen Symbols:
- Ten digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Six letters: A, B, C, D, E, F

Convention: $16_{10} = 10_{16} = 0\times 10$

Example: What is 0xA5 in base 10?
- $0xA5 = A5_{16} = (10 \times 16^1) + (5 \times 16^0) = 165_{10}$
Ques.on

Which of the following orderings is correct?

(A) 0b1010 < 0xC < 11
(B) 0xC < 0b1010 < 11
(C) 0b1010 < 11 < 0xC
(D) 11 < 0b1010 < 0xC
(E) 0xC < 11 < 0b1010
Question

Which of the following orderings is correct?

(A) \( 0b1010 < 0xC < 11 \)
(B) \( 0xC < 0b1010 < 11 \)
(C) \( 0b1010 < 11 < 0xC \)
(D) \( 11 < 0b1010 < 0xC \)
(E) \( 0xC < 11 < 0b1010 \)

\( 0b1010 = (8 \times 1) + (2 \times 1) = 10 \)
\( 0xC = 12 \)

10 < 11 < 12
Base Conversion
Converting to Base 10

Can convert from any base to base 10

- \(110_2 = (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 6_{10}\)
- \(0xA5 = A5_{16} = (10 \times 16^1) + (5 \times 16^0) = 165_{10}\)

We learned to think in base 10, so this is fairly natural for us.

Challenge: Convert into other bases (e.g. 2, 16)
Challenge Question

Convert $13_{10}$ to binary

Hints:

• $2^3 = 8$
• $2^2 = 4$
• $2^1 = 2$
• $2^0 = 1$
Converting from Decimal to Binary

Given a decimal number N:

- List increasing powers of 2 from right to left until $\geq N$
- From left to right, ask is that $(\text{power of 2}) \leq N$?
  - If **YES**, put a 1 below and subtract that power from N
  - If **NO**, put a 0 below and keep going

Example for 13:

<table>
<thead>
<tr>
<th>2^4=16</th>
<th>2^3=8</th>
<th>2^2=4</th>
<th>2^1=2</th>
<th>2^0=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \begin{array}{c}
\times5 \\
\times3 \\
\times2 \\
\times1 \\
\times0 \\
\end{array} \begin{array}{c} 13 \\
\underline{5} \\
\underline{1} \\
\underline{0} \\
\end{array} \]
Converting from Decimal to Base B

Given a decimal number N:

• List increasing powers of B from right to left until ≥ N
• From left to right, ask is that (power of B) ≤ N?
  • If **YES**, put *how many* of that power go into N and subtract from N
  • If **NO**, put a 0 and keep going

Example for 165 into hexadecimal (base 16):

<table>
<thead>
<tr>
<th>16^2=256</th>
<th>16^1=16</th>
<th>16^0=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A (10)</td>
<td>5</td>
</tr>
</tbody>
</table>
Converting Binary ↔ Hexadecimal

Hex → Binary

• Substitute hex digits, then drop leading zeros
• Example: 0x2D in binary
  • 0x2 is 0b0010, 0xD is 0b1101
  • Drop two leading zeros, answer is 0b101101

Binary → Hex

• Pad with leading zeros until multiple of 4, then substitute groups of 4
• Example: 0b101101
  • Pad to 0b 0010 1101
  • Substitute to get 0x2D
Binary → Hex Practice

Convert 0b100110110101101

• How many digits? 15
• Pad: 0b 0100 1101 1010 1101
• Substitute: 0x4DAD
Why are we learning this?

Why does all of this matter?

• Humans think about numbers in base 10, but...
  computers think about numbers in base 2
• How is it that computers can do all of the amazing things that they do?
  Binary encoding
Binary Encoding
Numerical Encoding

• AMAZING FACT: You can represent anything countable using numbers!
  • Need to agree on an encoding
  • Kind of like learning a new language

Examples:
• Decimal Numbers: 0→0b0, 1→0b1, 2→0b10, etc.
• English Letters: BJC→0x424A43, yay→0x796179
• Emoticons: 😊 0x0, 😞 0x1, 😊 0x2, 😊 0x3, 😈 0x4, 😍 0x5
Binary Encoding

With N binary digits, how many things can you represent?

- Need N bits to represent \( n \) things, where \( 2^N \geq n \)
- **Example:** 5 bits for alphabet because \( 2^5 = 32 > 26 \)

A binary digit is known as a **bit**
A group of 4 bits (1 hex digit) is called a **nibble**
A group of 8 bits (2 hex digits) is called a **byte**

**bit**\( \rightarrow \) 2 things, **nibble**\( \rightarrow \) 16 things, **byte**\( \rightarrow \) 256 things
So What’s It Mean?

A sequence of bits can have many meanings!
Consider the hex sequence 0x4E6F21
Common interpretations include:

• The decimal number 5140257
• The characters “No!”
• The background color of this slide
• The real number $7.203034 \times 10^{-39}$ [floating point]

It is up to the program/programmer to decide how to interpret the sequence of bits
Number Representation Recap

**Humans** think about numbers in decimal

**Computers** think about numbers in binary

Base conversion to go between

- Hex is more human-readable than binary

All information on a computer is in binary

- Nice because big difference between “high” and “low”

Binary encoding can represent *anything*!

- Program needs to know how to interpret bits
Operators Recap

• NOT: ~
  • This will flip all bits in the operand

• AND: &
  • This will perform a bitwise AND on every pair of bits

• OR: |
  • This will perform a bitwise OR on every pair of bits

• XOR: ^
  • This will perform a bitwise XOR on every pair of bits

• SHIFT: <<, >>
  • This will shift the bits right or left
    • logical vs. arithmetic
Operators Recap

• NOT: !
  • Evaluates the entire operand, rather than each bit
  • Produces a 1 if == 0, produces 0 if nonzero

• AND: &&
  • Produces 1 if both operands are nonzero

• OR: ||
  • Produces 1 if either operand is nonzero
Lab 1

• Worksheet in class

• Tips:
  • Work on 8-bit versions first, then scale your solution to work with 32-bit inputs
  • Save intermediate results in variables for clarity
  • Shifting by more than 31 bits is **UNDEFINED**. This will NOT yield 0
Examples

Create 0xFFFFFFFF using only one operator

• Limited to constants from 0x00 to 0xFF
• Naïve approach:
  0xFF + (0xFF << 8) + (0xFF << 16) ...
• Better approach:
  ~0x00 = 0xFFFFFFFF
Examples

Replace the leftmost byte of a 32-bit integer with 0xAB

• Let our integer be x
• First, we want to create a mask for the lower 24 bits
  • \( \sim (0xFF \ll 24) \) will do that using just two operators
• (x & mask) will zero out the leftmost 8 bits
• Now, we want to OR in 0xAB to those zeroed-out bits
• Final result:
  • \((x & mask) \mid (0xAB \ll 24)\)
• Total operators: 5