Announcements

1. Still grading the midterms; stay tuned
2. Gradescope for the homework problems is up and running!
Converting Hexadecimal to Binary

<table>
<thead>
<tr>
<th>bit pattern</th>
<th>hex digit</th>
<th>bit pattern</th>
<th>hex digit</th>
<th>bit pattern</th>
<th>hex digit</th>
<th>bit pattern</th>
<th>hex digit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0100</td>
<td>4</td>
<td>1000</td>
<td>8</td>
<td>1100</td>
<td>c or C</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>0101</td>
<td>5</td>
<td>1001</td>
<td>9</td>
<td>1101</td>
<td>d or D</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>0110</td>
<td>6</td>
<td>1010</td>
<td>a or A</td>
<td>1110</td>
<td>e or E</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>0111</td>
<td>7</td>
<td>1011</td>
<td>b or B</td>
<td>1111</td>
<td>f or F</td>
</tr>
</tbody>
</table>

So 0x401d2f1b is 0100 0000 0001 1101 0010 1111 0001 1011
Similarly, 0000 0000 0000 0000 0000 0001 0101 1001 is 0x00000159

Easiest way to do this:

• Binary to hexadecimal: group binary digits in sets of 4, starting at the end; then use the table to translate
• Hexadecimal to binary: translate each hexadecimal digit to the 4 corresponding digits in the table above and merge them
Decimal to Binary

• Repeatedly divide by 2, then stop when you get 0
• Record the remainders; those are the binary digits

Example: 345 in decimal:
• $345 / 2 = 172 \text{ r } 1$
• $172 / 2 = 86 \text{ r } 0$
• $86 / 2 = 43 \text{ r } 0$
• $43 / 2 = 21 \text{ r } 1$
• $21 / 2 = 10 \text{ r } 1$
• $10 / 2 = 5 \text{ r } 0$
• $5 / 2 = 2 \text{ r } 1$
• $2 / 2 = 1 \text{ r } 0$
• $1 / 2 = 0 \text{ r } 1$

• So 345 base 10 is 101011001 in base 2
Binary to Decimal

• Each digit is a power of 2, starting from the right (which is $2^0$)
• So 101011001 in base 2 is:
  • $1 \times 2^8 + 0 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 =$
  • $256 + 0 + 64 + 0 + 16 + 8 + 0 + 0 + 1 = 345$
Dealing with Bits: Operation Tables

<table>
<thead>
<tr>
<th>and (&amp;)</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Examples:
- \( 10 \& 01 = 00 \)
- \( 11 \& 01 = 11 \)

| or (|) | 0 | 1 |
|-------|---|---|
| 0     | 0 | 0 |
| 1     | 0 | 1 |

Examples:
- \( 10 \mid 01 = 11 \)
- \( 11 \mid 01 = 11 \)

<table>
<thead>
<tr>
<th>xor (^)</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Examples:
- \( 10 \mid 01 = 11 \)
- \( 11 \mid 01 = 11 \)

<table>
<thead>
<tr>
<th>not (~)</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Examples:
- \( \sim 10 = 01 \)
- \( \sim 11 = 00 \)
Dealing with Bits: Shift Operations

• \( b \ll n \): shift \( b \) left \( n \) bits
  • Bits shifted beyond the end of the word are discarded
  • 0 bits are inserted at the right

• \( b \gg n \): shift \( b \) right \( n \) bits
  • If \( b \) is signed, the high-order (most significant) bit is propagated
  • If \( b \) is unsigned, 0 bits are inserted at the left
  • Bits shifted beyond the end of the word are discarded
How to Extract Bits

• Number the bits from 31 to 0
• To get the \(i\)-th bit of unsigned int \(x\):

\[
b = (x>>i) & 01
\]

Example: 345 in binary:

\[
0000 0000 0000 0000 0000 0001 0101 1001
\]

Extract bit 8:

\[
b = (345 >> 8) & 01 =
\]

\[
(0000 0000 0000 0000 0000 0000 0001 0101 1001 >> 8) & 01 =
\]

\[
(0000 0000 0000 0000 0000 0000 0000 0001) & 01 = 1
\]
How to Extract Groups of Bits

• Number the bits from 31 to 0
• To get the i-th through j-th bits of unsigned int x:
  \[ b = (x >> j) \& 0xZ \]
  where Z is the hex representation of i–j bits

Example: 345 in binary:
0000 0000 0000 0000 0000 0001 0101 1001

Extract bits 8 to 5:

\[ b = (345 >> 5) \& 0xf = (345 >> 5) \& 0xf = \]
\[ (0000 0000 0000 0000 0000 0000 0000 0101) \& 0xf = 0101 = 5 \]
gdb

• To print in binary, use
• (gdb) print/t x