Introduction

In this activity you will review the material on integers, binary, and floating-point necessary for datalab. This activity was based on material developed by Professor Saturnino Garcia of the University of San Diego. It is used here with permission.

Each activity is designed to be solved in groups and take approximately 10 minutes.

Activity 1: Bit-level and Logical

1. De Morgan’s Law enables one to distribute negation over AND and OR. \( \neg(x \land y) = \neg(x) \lor \neg(y) \)
   Verify that De Morgan’s Law works for the following expression.

   \[
   \begin{array}{c|c|c|c|c}
   x & y & \neg(x \land y) & \neg(x) \lor \neg(y) \\
   0x5 & 0x7 & & \\
   \end{array}
   \]

   This section will explore logical operations. These operations contrast with bit-level in that they treat the entire value as a single element. In other languages, the type of these values would be termed, “bool” or “boolean”. C does not have any such type. Instead, the value of 0 is false and all other values are true.

   The three operators are AND (\&\&), OR (\|\|), and NOT (!). “!” is commonly termed “bang”.

   2. Evaluate the following expression: \((0x3 \&\& 0xC) = !((0x3 \& 0xC))\)

Activity 2: Shifts, Negation and Conditional

1. With 4-bit integers, what is the binary for -2? After right shifting by 1, what value(s) might we have?

2. Fill in the following table, assuming you only have 4 bits to represent the 2s complement integer.

   \[
   \begin{array}{c|c|c|c}
   x & \text{x in binary} & \neg x \text{ in binary} \\
   1 & & \\
   2 & & \\
   7 & & \\
   -8 & & \\
   \end{array}
   \]
1. Find an algorithm for computing the expression \((\text{cond}) \ ? \ t \ : \ f\), which equals \(t\) if \(\text{cond}\) is 1 and \(f\) if \(\text{cond}\) is 0.

```c
int conditional(int cond, int t, int f) {
    /* Compute a mask that equals 0x00000000 or 0xFFFFFFFF depending on the value of cond */
    int mask = ______________________________________;
    /* Use the mask to toggle between returning t or returning f */
    return __________________________________________;
}
```

Activity 3: Floating-point

1. How many representations for zero are there with denormalized floats? Are any of these representations the same for zero as an integer?

2. Which is larger, \(2^{127}\) or \(+\text{inf}\)? Does this ordering hold when these numbers are floats (i.e., if just the bit patterns are compared)?

3. What is the numerical form for these floating point encodings?

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Numerical Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>010001100110110110110100000000000</td>
<td>000000000001110000000000000000000</td>
</tr>
</tbody>
</table>

4. There are several possible rounding schemes for floating point values. There are two components of rounding. First, is what to do in general? Should the float be rounded up, down, to zero, or to nearest? The second component is what to do about ties with round to nearest. So should 9/2.0 be 4 or 5? The default IEEE scheme is round to nearest even. Apply it to the following values for a system that only has three bits for the fractional component of the final value, so final binary value should be \(1.xyz\).

<table>
<thead>
<tr>
<th>Value</th>
<th>Binary</th>
<th>Rounded</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 3/32</td>
<td>1 5/32</td>
<td>1 3/16</td>
<td>1 5/8</td>
</tr>
</tbody>
</table>
Activity 4: Divide and Conquer

Let’s count how many bits are set in a number. For each challenge, you can use any allowed operator allowed in the integer problems in datalab. Using 1 op, return the number of bits set in a 1-bit number. int bitCount1bit(int x) {return x;}

1. How about if there are two bits in the input? (4 ops max)

int bitCount2bit(int x)
{
    int bit1 = ___________
    int bit2 = ___________
    return ___________ + ___________ ;
}

2. How about if there are four bits? (8 ops max)

int bitCount4bit(int x)
{
    int mask = ___________

    int halfSum = ___________

    int mask2 = ___________

    return ___________ + ___________ ;
}

3. How about if there are eight bits? (12 ops max)

int bitCount8bit(int x)
{
    int mask = ___________

    int quarterSum = ___________

    int mask2 = ___________

    int halfSum = ___________

    int mask3 = ___________

    return ___________ + ___________ ;
}