15213 - Recitation 3 - Datalab

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. Given the following expression, complete the following table to verify for the 4-bit inputs. ~(x & y) == (~x) | (~y)

0011 1100

x	y	~(x & y)	(~x) | (~y)
0xF	0x1	1110	1110
0x5	0x7	1010	1010
0x3	0xC	1111	1111

1111 0001

0000 1110

0101 0111

0101 1100

12.

8+4.

0011 1100.

0000

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".

1. Evaluate the following expression: (0x3 && 0xC) == (0x3 & 0xC) X

0011
1100.

2. Test whether !!X == X holds across different values of X. Do the same for bitwise complement.

0000

X	!X	!!X	~X	~~X
-1	0	1	0	-1
0	1	0	-1	0
1	0	1	-2	1
2	0	1	-3	2

000000001

1111111110.

00— 10

11111101.

Activity 2: Shifts, Negation and Conditional

1. Suppose we right shift the value of "-2" by 1. What value do we expect? -1.

11111110

2. With 4-bit integers, what is the binary for -2? After right shifting by 1, what value(s) might we have?
   1110
   -1. 1111

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

x	x in binary	-x in binary
1	0001	1111
2	0010	1110
7	0111	1001
-8	1000	-

0111

1101

1111

1110

0111

11 1000

4. Find an algorithm for computing the expression `(cond) ? t : f`, which equals `t` if cond is 1 and `f` if cond is 0.

```
int conditional(int cond, int t, int f) {
 /* Compute a mask that equals 0x00000000 or
 0xFFFFFFFF depending on the value of cond */

 int mask = _(cond <<31) >>31________________________;

 /* Use the mask to toggle between returning t or returning f */

 return __(mask AND t) OR (mask AND f)______;
}
```

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?
   +0 and -0 Yes

2. Which is larger, 2^{127} or $+inf$? Does this ordering hold when these numbers are floats (i.e., if just the bit patterns are compared)?
   0b0, 1111 1110, 0.... AND 0b0, 1111 1111, 0....

3. 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 final values.

Value	Binary	Rounded	Final
1 3/32	1.00011	11	1.001
1 5/32	1.00101	01	1.001
1 7/8	1.111	-	1.111
1 5/8	1.101	-	1.101

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)

1010.

```
int bitCount2bit(int x)
{
  int bit1 = 1 & x___________;
  int bit2 =(2 & x) >>1___________;
  return ___bit1________ + ___bit2________ ;
}
```

0 1.
0 1
1 0.

0000+ 0101

0101+

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

1010

0011 x 0

```
int bitCount4bit(int x)
{
  int mask =0b0011___________;

  int halfSum = _bitCount2bit;(x & mask)

  int mask2 = 0b1100________;

  return halfSum________ + bitCount2bit((x & mask2)>>2)
}
```

0101

0000 XOR 0101

0101.

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

```
int bitCount8bit(int x)
{
  int mask = 0x0F________;

  int quarterSum = bitCount4bit;(mask & x)

  int mask2 = 0xF0_________;

  int halfSum = ___________;

  int mask3 = ___________;

  return ___________ + ___________ ;
}
```