Lecture 3 Activity Solution

Model 0: Review of Addition / Positive

1. 10110
2. 5 bits are required.
3. The number of bits in the result is one more than the number of bits of the operands.
4. You could truncate overflow bits, resulting in 0110.

Model 0: Review of Negative Integers

1. The leftmost bit in a non-negative number in two's complement is 0.
2.

Bits	Most Positive	Most Negative
1	0	-1
2	1	-2
3	3	-4
4	7	-8

3. $2^{N-1} - 1$
4. $-(2^{N-1})$
5. 10011111. If the two numbers are unsigned, the result is correct ($1111000_2 = 120_{10}, 0100111_2 = 39_{10}, 10011111_2 = 159_{10}$), but the result is not correct for signed numbers ($1111000_2 = -8_{10}, 0100111_2 = 39_{10}, 10011111_2 = -97_{10}$).
6. No, but the difference in expected results for signed integers comes from improper handling of overflow or sign extension.

Model 1: Bit-Level Operations

1.
   - 0x3501
   - 0xC3C3
   - 0xFFFF
2.

OP0	OP1	AND	OR	XOR
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	1	1	0

3.

Dec	Bin	X & 0x1
-2	1110	0000
-1	1111	0001
0	0000	0000
1	0001	0001
2	0010	0000

4. The decimal numbers -1 and 1, which both are odd and therefore have a 1 in the rightmost (least-significant) bit.

5.
```
for each bit in X:
  if that bit is set in FLAG but not set in X:
    return false
return true
```

6. The OR (|) operation is setting the relevant bits in the file access mode to create a flag with the bits set for all of `O_WRONLY`, `O_CREAT`, and `O_TRUNC`.

7.

x	y	∼(x & y)	(∼x) | (∼y)	equal?
0xF	0x1	1110	1110	Y
0x5	0x7	1010	1010	Y
0x3	0xC	1111	1111	Y

Model 2: Logical Operations

1. 1 value is false and 15 values are true.

2. 0x3 && 0xC = 0001, 0x3 & 0xC = 0000, so 0x3 && 0xC == 0x3 & 0xC is false.

3.

X	!X	!!X	!!X == X
-1	0	1	0
0	1	0	1
1	0	1	1
2	0	1	0

4. Yes, the results differ—every ∼∼X = X. Note that ∼∼X is a no-op (gives X back for all X) while !!X is not.

Model 2: Multiplication and Division

1.

Value	<<	Result
0x30	1	0x60
0x5A	4	0x5A0
0x11D	31	0x80000000

2. $X = 6_{10} = 0110_2$

3. Two acceptable answers: x << 2 + x << 1, or (x + x + x) << 1.

4. The largest 3-bit unsigned integer is $111_2 = 7_{10}$, its value squared is 49, which requires 6 bits.

5. $001_2 = 1_{10}$, if truncating excess bits.

6.

Value	>>	Result
0x30	1	0x18
0x5A	4	0x5
0x11	3	0x2

7.

Value	>>	Result
48	1	24
90	4	5
17	3	2

A single right shift is equivalent to dividing by 2, so right shifting by N is equivalent to dividing by 2^N.

8. 0xA >> 1 = 0x5

9. We expect that -2 >> 1 = -1.

10. $-2_{10} = 1110_2$ in two's complement. After right shifting by 1, we get $0111_2 = 7_{10}$.

11. We could replicate the most significant (leftmost) bit so all bits shifted "in" would be copies of the leftmost bit instead of zeroes.

12.
```
while (x != 0)
{
  saveNextBit(x & 0x1);
  x = x >> 1;
}
```