Review of Binary Math

- Each digit of a decimal number represents a power of 10:

$$\mathbf{258} = 2\text{x}10^2 + 5\text{x}10^1 + 8\text{x}10^0$$

- Each digit of a binary number represents a power of 2:

$$\mathbf{01101_2} = 0\text{x}2^4 + 1\text{x}2^3 + 1\text{x}2^2 + 0\text{x}2^1 + 1\text{x}2^0$$
$$\mathbf{= 13_{10}}$$

Decimal to Binary Conversion

- Let's say I give you number 11 in decimal. How would you represent this in binary?
  - Keep dividing by 2 and write down the remainders!

11 in decimal is 1011 in binary!

Use the quotient from previous row.

Number	Quotient = Number / 2	Remainder = Number % 2	
11			

Decimal to Binary Conversion

- Let's say I give you number 11 in decimal. How would you represent this in binary?
  - Keep dividing by 2 and write down the remainders!

11 in decimal is 1011 in binary!

Use the quotient from previous row.

Number	Quotient = Number / 2	Remainder = Number % 2	
11	5	1	Least Significant Bit
5	2	1	
2	1	0	
1	0	1	Most Significant Bit

Hexadecimal Numbers

- Base 16 numbers, where valid values are:
  - 0 to 9 as in decimal, and
  - 10 is A
  - 11 is B
  - ..
  - 15 is F

Hex numbers are typically expressed as 0x____

- Writing a binary number in hex(-adecimal):
  - 0000010111111010 = 0000 0101 1111 1010 = 0x05fa
  - In Verilog (more about this in the handout of Lab 3):
    - 16’b0000_0101_1111_1010
    - 16’h05FA (16’h05fa is fine too)

Two's complement

- Need to know how to get 1's complement:
  - Given number `X` with `n` bits, take $(2^n-1)-X$
  - Negates each individual bit (bitwise NOT).

```
01001101   →   10110010
11111111   →   00000000
```

- 2's complement = (1's complement + 1)

```
01001101   →   10110011
11111111   →   00000001
```

Know this!

- Note: Adding a 2's complement number to the original number produces a result of zero.

Signed subtraction

- Negative numbers are generally stored in 2's complement notation.
  - Reminder: 1's complement → bits are the bitwise NOT of the equivalent positive value.
  - 2's complement → one more than 1's complement value; results in zero when added to equivalent positive value.
    - Subtraction can then be performed by using the binary adder circuit with negative numbers.

Signed representations

Decimal	Unsigned	Signed 2's
7	111	---
6	110	---
5	101	---
4	100	---
3	011	011
2	010	010
1	001	001
0	000	000
-1	---	111
-2	---	110
-3	---	101
-4	---	100

Rules about signed numbers

- When thinking of signed binary numbers, there are a few useful rules to remember:
  - The largest positive binary number is a zero followed by all ones.
  - The binary value for -1 has ones in all the digits.
  - The most negative binary number is a one followed by all zeroes.

- There are 2^n possible values that can be stored in an n-digit binary number.
  - 2^{n-1} are negative, $2^{n-1}-1$ are positive, and one is zero.

  - For example, given an 8-bit binary number:
    - There are 256 possible values
    - One of those values is zero
    - 128 are negative values (11111111 to 10000000)
    - 127 are positive values (00000001 to 01111111)

-1 to -128

1 to 127

Practice 2's complement!

- Assume 4-bits signed representation!
- Write these decimal numbers in binary:
  - 2 => 0010
  - -1 => 1111
  - 0 => 0000
  - 8 => Not possible to represent!
  - -8 => 1000
- What is max positive number? => 7 (i.e., 2^{4-1} -1)
- What is min negative number? => -8 (i.e, -2^{4-1})

Sign & Magnitude Representation

- The Sign part: one bit is designated as the sign (+/-).
  - `0` for positive numbers
  - `1` for negative numbers
- The Magnitude part: Remaining bits store the positive (i.e., unsigned) version of the number.

- Example: 4-bit binary numbers:
  - `0110` is `6` while `1110` is `–6` (most significant bit is the sign)
  - What about `0000` and `1000`? => zero (two ways)
- Sign-magnitude computation is more complicated.
  - 2's complement is what today's systems use!