Excess Notation

- **This fixed length notation (i.e., the length of the bit pattern used can not be altered once set at the beginning) makes it possible to store negative (-) and non-negative (+ including zero) values by treating the left-most digit referred to as the *Most Significant Bit* (MSB) as representing the sign of the number.**
- **In excess notation the MSB serves as the *sign bit* - a 1 represents the non-negative (+) sign and a 0 indicates a negative (-) number.**

Note the two examples below.

Example # 1.

In the case of a 4-bit pattern, for example: 0110 the digit/column value of the most significant bit is 8, so 4 bit patterns are referred to as an excess (8) notation.

- **To convert this example find the sum value of the entire pattern as though a standard binary number:**

  (0x8) + (1x4) + (1x2) + (0x1) = 6_{10}

- **Then subtract the excess value,8, from the sum, (6 8)**
- **The result is a signed value, -2.**

Example # 2.

In the case of a 5-bit pattern example, 11110, the digit/column value of the most significant bit is 16, so 5-bit patterns are referred to as an excess (16) notation.

- **To convert this example find the sum value of the entire pattern as though a standard binary number:**

  (1x16) + (1x8) + (1x4) + (1x2) + (0x1) = 16 + 8 + 4 + 2 + 0 = 30

- **Then subtract the current excess value, 16, from the sum, (30 - 16)**
- **The result is a signed value, + 14.**
- **Therefore, it is evident that in excess notation, the sign bit of 0 represents the negative sign and 1 represents the non-negative sign to denote a signed value.**