Encoding Integers in Binary Logic

Abyan Majid

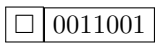
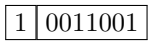
July 5, 2023

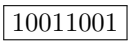
For the sake of simplicity, I will only be encoding 8-bit integers in this note.
Of course, the encoding methods in this note is also applicable to higher-bit integers.

1 Size & magnitude notation

In size & magnitude notation, the leftmost bit represents the sign ("0" means positive, "1" means negative), meanwhile the remaining bits represents the magnitude of the integer.

Suppose we want to translate the integer "-25" to binary.

1. First, we get the magnitude of -25 (which is just the binary of its natural, or $|-25|$).
   
2. Second, Make the leftmost bit = 1.
   

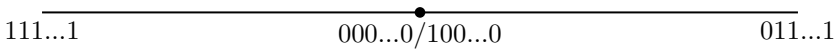
So, the 8-bit binary of "-25" is 

1.0.1 Two encoding of 0

This method is much more intuitive than 2s complement, but it has an undesirable property which is that zero has two possible encodings which are:

- 000...0
- 100...0

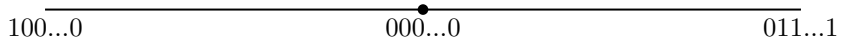
This is because 0 is neither positive or negative.



And that is why this method is less desirable. In all circumstances, you should encode integers in 2s complement notation.

2 2s complement notation (Preferred Method)

In 2s complement notation, we discard the undesirable property that 0 has two encodings by letting 0 be equal to only 000...0 and inverting the bits for every negative integer.



To translate a positive integer, you treat it like a natural and get the binary. On the other hand, to translate a negative integer, say "-25", you follow these steps:

1. Get the binary representation of the integer
   $\boxed{00011001}$
2. Invert all the bits to get the 1s complement (so you turn 0s to 1s, and 1s to 0s)
   $\boxed{00011001} \rightarrow \boxed{11100110}$ (this is the 1s complement)
3. Add 1 to the 1s complement

$$\begin{array}{r} 11100110 \\ +1 \\ \hline 11100111 \end{array}$$

So, the 8-bit binary of "-25" is $\boxed{11100111}$.

Furthermore, this method also allows us to consistently perform basic arithmetic between binary numbers. Suppose we want to add 10 to -25,

$$\begin{array}{r} -25 \\ +10 \\ \hline -15 \end{array}$$

But in binary logic. So, given that the 8-bit binary of 10 is 00001010, we have:

$$\begin{array}{r} 11100111 \\ +00001010 \\ \hline 11110001 \end{array}$$

The 8-bit binary of -15 is indeed 11110001, so this proves that the 2s complement notation allows for arithmetic operations between binary numbers.