A simple derivation.

Suppose we want to represent arbitrary-precision integers, namely integers that may be too large to be stored in any one machine word. (E.g., in Java, `java.util.BigDecimal` serves that purpose.) Let \( n \) be such integer. We can represent the binary expansion of \( n \) by splitting the sequence of bits into a sequence of words. For example, if we use 4-bit words (a.k.a. nibbles), and \( n = (20)_{10} = (10100)_{2} \), then we would need two words: \([0001], [0100] \). Thus, if the binary expansion of \( n \) contains \( k \) bits, and we use \( l \)-bit words, we must allocate \( \lceil k/l \rceil \) words. (In the previous example, \( k = 5 \), \( l = 4 \), and we required \( \lceil k/l \rceil = 2 \) words.) However, what if the binary expansion of \( n \) is not given, can we use some formula to compute the fewest number of bits \( n \) would require? The answer is affirmative. In the ensuing discussion, assume that \( n \) is any positive integer \( \geq 1 \), and \( b \) is any base \( \geq 2 \). Fortunately, mathematics can deal with infinitely large numbers allowing one to derive formulas that even hold for those \( n \) that would require the space of the whole universe to store them.

Let \( x \) be the fewest number of \( b \)-digits required to represent \( n \) in base \( b \). The largest number that can be represented with \( x \) digits is \( b^x - 1 \). Therefore, \( n \leq b^x - 1 \). Since \( x \) is the smallest, \( b^{x-1} \leq n \). (Otherwise, \( n < b^{x-1} \) which implies \( n \leq b^{x-1} - 1 \), and thus \( x - 1 \) suffices to represent \( n \).) Combining the two inequalities relating \( n \) and \( x \), we obtain

\[
b^{x-1} \leq n \leq b^x - 1
\]

Using the fact that \( n \leq b^x - 1 \) implies \( n < b^x \),

\[
b^{x-1} \leq n < b^x
\]

Since \( \log_b \) is a monotonic\(^1 \) function we can apply it to each side of the inequality

\[
\log_b b^{x-1} \leq \log_b n < \log_b b^x
\]

Using the fact that \( \log_b b^x = x \), for any \( b > 0 \), \( x > 0 \),

\[
x - 1 \leq \log_b n \leq x
\]

Since \( +1 \) is a monotonic function, we can apply it to each side,

\[
x \leq \log_b(n) + 1 < x + 1
\]

Now observe that the left and right sides of the inequality are integers whereas the middle is a real. Let \( x = [\log_b(n) + 1] \) and observe that it satisfies the following (where we replaced all occurrences of \( x \) in (1) with \( [\log_b(n) + 1] \)),

\[
[\log_b(n) + 1] \leq \log_b(n) + 1 < [\log_b(n) + 1] + 1
\]

\(^1\)A function, \( f \), is monotonic iff for any \( x, y \) such that \( x \leq y \), \( f(x) \leq f(y) \).
where \( |y| \), for any real \( y \), is formally defined as \( |y| = \max\{n \in \mathbb{Z} \mid n \leq y\} \), and is called the floor function or the greatest integer less than or equal to \( y \). It is easily seen that

\[ x = \lceil \log_b(n) + 1 \rceil = \lceil \log_b n \rceil + 1 \]

is a unique solution to (1).

To apply the formula to our example, let \( b = 2 \), \( n = 20 \) and recall that \( \log_b x = \log_{10} x / \log_{10} b \). Then, \( \lceil \log_2 20\rceil + 1 = \lceil 4.321\ldots \rceil + 1 = 5 \) is the fewest number of bits required to represent 20.