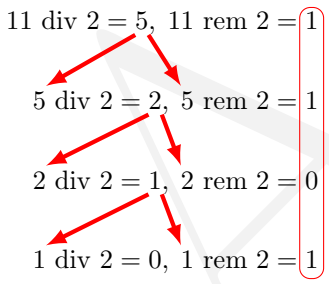
5.7 Programming intermezzo

Conversion of integers between decimal and binary form is a key concept in order to understand some of the basic properties of calculations on the computer. From binary to decimal is straight forward using the power-of-two algorithm, i.e., given a sequence of $n+1$ bits that represent an intager $b_n b_{n-1} \ldots b_0$, where b_n and b_0 are the most and least significant bits, then the decimal value is calculated as,

$$v = \sum_{i=0}^{n} b_i 2^i \tag{5.1}$$

For example $10011_2 = 1 + 2 + 16 = 19$. From decimal to binary is a little more complex, but a simple divide-by-two algorithm exists. The key to understanding the divide-by-two algorithm is to realize that when you divide a number by two, then that is equivalent to shifting its binary representation 1 to the right. E.g., $10 = 1010_2$ and $10/2 = 5 = 110_2$. Odd numbers have $b_0 = 1$, e.g., $11_{10} = 1011_2$ and $11_{10}/2 = 5.5 = 101.1_2$. Hence, if we divide any number by two and get a non-integer number, then its least significant bit was 1. Another way to express this is that the least significant bit is the remainder after integer division by two. Sequential application of this idea leads directly to the divide-by-two algorithm. E.g., if we were to convert the number 11_{10} on decimal form to binary form we would perform the following steps:



Here we used div and rem to signify the integer division and remainder operators. The algorithms stops, when the result of integer divsion is zero. Reading off the remainder from below and up we find the sequence 1011_2, which is the binary form of the decimal number 11_{10}. For integers with a fractional part, the divide-by-two may be used on the whole part, while multiply may be used in a similar manner on the fractional part.