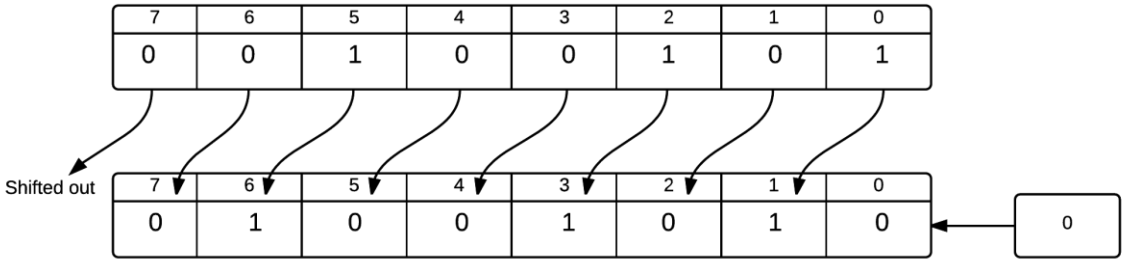
Multiplication and division operations are slow for a CPU to process. **Shifting bits** in a **register** to the left or right is fast. As the CPU uses **binary**, you can replace multiplication and division by shifting bits if they are to the power of 2.

Binary shifts allow us to move all the bits stored in a **register** on the processor either left or right. If we move the bits to the left then it is a **left-shift**, if we move the bits to the right then it is a **right-shift**.

Left-shift

In a **left-shift** we move the bits to the left. This leaves an empty space at the right-most bit which is replaced with a zero. The left-most bit is **shifted out** of the register. The following example shows how the number 00100101 will be left-shifted to the number 01001010.



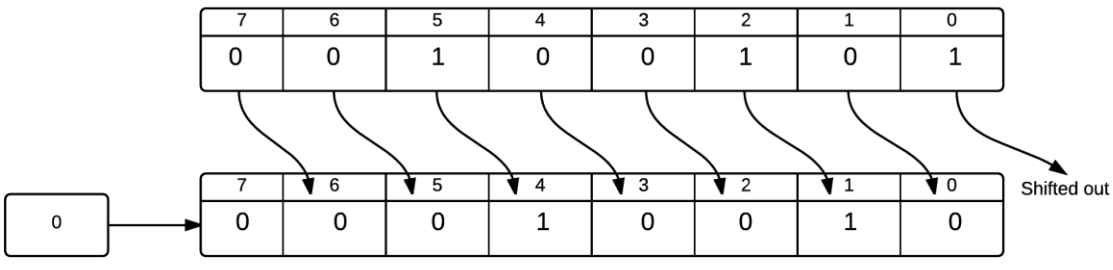
We start with the number 00100101 which is 37 in decimal. The result of the left-shift is 01001010 which is 74 in decimal. Because each column in binary has a value 2 times greater than the previous column, the effect of a left-shift is to multiply by 2.

Although if there is a 1 in the left-most bit when we perform a left-shift it will **overflow** and the bit will be lost.

Right-shift

Right-shift works exactly the same as left-shift except all the bits are moved to the right. If there is a 1 in the right-most bit then it will be shifted out.

Performing a right-shift is the same as dividing a binary number by 2.



The example above starts with 00100101 which is 37 in decimal. It ends with 00010010 after the right-shift, which is 18 in decimal. 37÷2 = 18.5. The right bit was shifted out and this would be responsible for the 0.5 difference in the results. This is known as an **underflow**.