1/11/2020 2.13. Binary

## 2.13 Binary

Normally, a programmer can think in terms of base ten numbers. However, a compiler must allocate some finite quantity of bits (e.g., 32 bits) for a variable, and that quantity of bits limits the range of numbers that the variable can represent. Thus, some background on how the quantity of bits influences a variable's number range is helpful.

Because each memory location is composed of bits (0s and 1s), a processor stores a number using base 2, known as a **binary number**.

For a number in the more familiar base 10, known as a **decimal number**, each digit must be 0-9 and each digit's place is weighed by increasing powers of 10.

Table 2.13.1: Decimal numbers use weighed powers of 10.

Decimal number with 3 digits	Representation				
212	$= 2 \cdot 10^{2}$ $= 2 \cdot 100$ $= 200$ $= 212$	$+ 1 \cdot 10^{1}  + 1 \cdot 10  + 10$	$+  2 \cdot 10^{0} \\ +  2 \cdot 1 \\ +  2$		

Feedback?

In **base 2**, each digit must be 0-1 and each digit's place is weighed by increasing powers of 2.

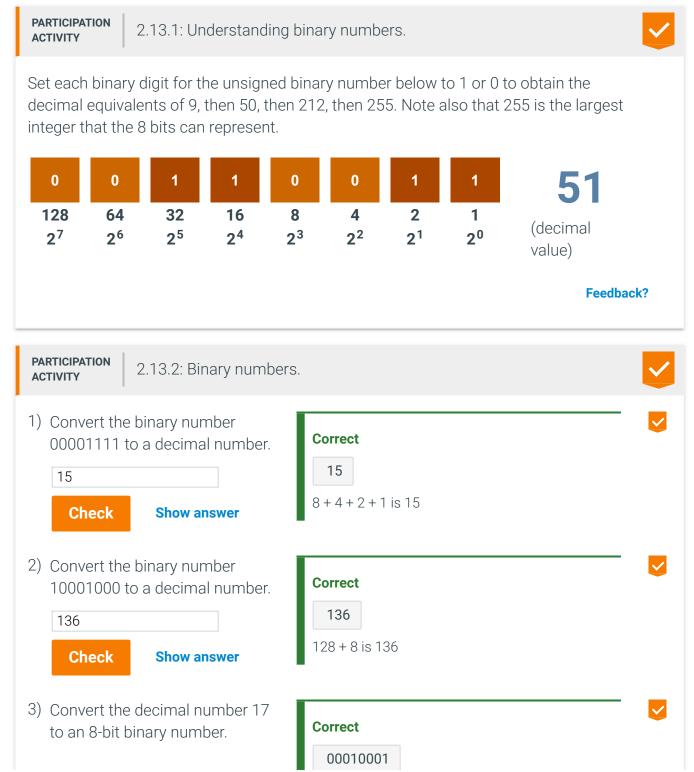
Table 2.13.2: Binary numbers use weighed powers of 2.

Binary number with 4 bits	Representation				
1101	$= 1 \cdot 2^{3}$ $= 1 \cdot 8$ $= 8$ $= 13$	$+  1 \cdot 2^2 \\ +  1 \cdot 4 \\ +  4$	$+ 0 \cdot 2^{1}  + 0 \cdot 2  + 0$	$+ 1 \cdot 2^{0} \\ + 1 \cdot 1 \\ + 1$	

1/11/2020 2.13. Binary

Feedback?

The compiler translates decimal numbers into binary numbers before storing the number into a memory location. The compiler would convert the decimal number 212 to the binary number 11010100, meaning 1\*128 + 1\*64 + 0\*32 + 1\*16 + 0\*8 + 1\*4 + 0\*2 + 0\*1 = 212, and then store that binary number in memory.



1/11/2020 2.13. Binary

