rather than directly subtracting the subtrahend from the minuend. These are, of course, identical operations: A – B = A + (–B). This type of arithmetic is referred to as subtraction by addition of the *two's complement*. The two's complement is the negative representation of a number that allows the identity A – B = A + (–B) to hold true.

Subtraction requires a means of expressing negative numbers. To this end, the most-significant bit, or left-most bit, of a binary number is used as the sign-bit when dealing with signed numbers. A negative number is indicated when the sign-bit equals 1. Unsigned arithmetic does not involve a sign-bit, and therefore can express larger absolute numbers, because the MSB is merely an extra digit rather than a sign indicator.

The first step in performing two's complement subtraction is to convert the subtrahend into a negative equivalent. This conversion is a two-step process. First, the binary number is inverted to yield a *one's complement*. Then, 1 is added to the one's complement version to yield the desired two's complement number. This is illustrated below:

	0	1	0	1	Original number (5)
	1	0	1	0	One's complement
+	0	0	0	1	Add one
	1	0	1	1	Two's complement (–5)

Observe that the unsigned four-bit number that can represent values from 0 to 15_{10} now represents signed values from –8 to 7. The range about zero is asymmetrical because of the sign-bit and the fact that there is no negative 0. Once the two's complement has been obtained, subtraction is performed by adding the two's complement subtrahend to the minuend. For example, 7 – 5 = 2 would be performed as follows, given the –5 representation obtained above:

1	*1*	*1*	*1*	*0*	*Carry bits*
	0	1	1	1	Minuend (7)
+	1	0	1	1	"Subtrahend" (–5)
	0	0	1	0	Result (2)

Note that the final carry-bit past the sign-bit is ignored. An example of subtraction with a negative result is 3 – 5 = –2.

		1	*1*	*0*	*Carry bits*
	0	0	1	1	Minuend (3)
+	1	0	1	1	"Subtrahend" (–5)
	1	1	1	0	Result (–2)

Here, the result has its sign-bit set, indicating a negative quantity. We can check the answer by calculating the two's complement of the negative quantity.

	1	1	1	0	Original number (–2)
	0	0	0	1	One’s complement
+	0	0	0	1	Add one
	0	0	1	0	Two’s complement (2)

This check succeeds and shows that two’s complement conversions work “both ways,” going back and forth between negative and positive numbers. The exception to this rule is the asymmetrical case in which the largest negative number is one more than the largest positive number as a result of the presence of the sign-bit. A four-bit number, therefore, has no positive counterpart of –8. Similarly, an 8-bit number has no positive counterpart of –128.

1.7 MULTIPLICATION AND DIVISION

Multiplication and division follow the same mathematical rules used in decimal numbering. However, their implementation is substantially more complex as compared to addition and subtraction. Multiplication can be performed inside a computer in the same way that a person does so on paper. Consider 12 × 12 = 144.

		1	2	
	X	1	2	
		2	4	Partial product × 10^0
+	1	2		Partial product × 10^1
	1	4	4	Final product

The multiplication process grows in steps as the number of digits in each multiplicand increases, because the number of partial products increases. Binary numbers function the same way, but there easily can be many partial products, because numbers require more digits to represent them in binary versus decimal. Here is the same multiplication expressed in binary (1100 × 1100 = 10010000):

				1	1	0	0	
			X	1	1	0	0	
				0	0	0	0	Partial product × 2^0
			0	0	0	0		Partial product × 2^1
		1	1	0	0			Partial product × 2^2
+	1	1	0	0				Partial product × 2^3
1	0	0	1	0	0	0	0	Final product