Representations of Integers

Carol Zander

In this section, we will only consider representations of integers. We are concerned about how they are stored and manipulated in a computer.

In everyday life, we use the **decimal system, base 10,** which uses the digits 0 through 9, the symbols: 0 1 2 3 4 5 6 7 8 9 . The use of alternative number systems is important in computer science. Computers don't use base 10 when they process numbers and code; they use binary digits (or bits), which can only be 0 or 1, instead of 0 through 9 in the decimal system. To understand how other number systems work, consider how the decimal system works. First, don't think of ten as ten. The base is always represented by one zero, 10. Consider **4597**, by which we mean four thousand, five hundred, and ninety-seven. For convenience, we start at the rightmost digit, so **4597** can be thought of as the sequence $\mathbf{s_0=7,\ s_1=9,\ s_2=5,\ s_3=4}$. This is converted into a number by taking each digit and multiplying it by a power of ten (symbol 10 in decimal). If n is the number of digits in our number, then we can compute the number from the sequence s using:

$$\sum_{i=0}^{n-1} s_i 10^i = 7\cdot 10^0 + 9\cdot 10^1 + 5\cdot 10^2 + 4\cdot 10^3 = 4597$$

In a manner more familiar, a number 4597 can be viewed as

4	5	9	7
|	|	|	|
$4\cdot 10^3$	$5\cdot 10^2$	$9\cdot 10^1$	$7\cdot 10^0$ (or $7\cdot 1$)

If you think of the digits as just symbols, then this pattern can be used to represent numbers of any base. The common bases used in computing are the binary number system (base 2), the octal number system, (base 8), and the hexadecimal number system (base 16, if we speak in decimal). In terms of a sequence, working in another number system is the same, except a different base for the exponent is used. In **binary**, write:

$$\sum_{i=0}^{n-1} s_i 2^i$$

So, **101101 (base 2) is $1\cdot 1 + 0\cdot 2 + 1\cdot 4 + 1\cdot 8 + 0\cdot 16 + 1\cdot 32 = 45$**. How do we **write 23 in binary**? First, we have to find the largest digit. We find the largest power of 2 that is smaller than (or equal to) the number. **$2^4 = 16$** is smaller than 21, but $2^5 = 32$ is not. This means that our number will have 5 digits. In binary, the digit can only be 0 or 1, so in some ways it is a little easier than decimal (or other number systems). The first digit is, of course, 1. We then remove 16 from the number (23-16) to get 7 and go to the next digit. The next digit is for $2^3 = 8$. The amount left in the number is smaller than 8, so the next digit is 0. At the $2^2 = 4$ spot, we have a 1, since 7 is larger than 4, leaving 3 after removing the four. There is a 1 at the 2 position. After subtracting 2 from 3, there is a 1 at the final position, so the result is: **10111_2**.

Another approach is to continually divide by 2 using grade school long division:
$2\sqrt{}23$ is 11 with remainder of 1
$2\sqrt{}11$ is 5 with remainder of 1
$2\sqrt{}5$ is 2 with remainder of 1
$2\sqrt{}2$ is 1 with remainder of 0
$2\sqrt{}1$ is 0 with remainder of 1 The remainders backwards are your number in base 2: 10111

In the **octal system**, **base 8**, the symbols 0, 1, 2, 3, 4, 5, 7 are used. The base, the number eight ,is represented as seen previously by 10. For example, consider 1234_8 :

1	2	3	4		
|	|	|	|		
$1\cdot 8^3$	$2\cdot 8^2$	$3\cdot 8^1$	$4\cdot 8^0$	=	$512 + 128 + 24 + 4 = 668_{ten}$

Using the division technique, convert the number 4597 in base ten to base 8:
$8\sqrt{4597}$ is 574 with remainder of 5
$8\sqrt{574}$ is 71 with remainder of 6
$8\sqrt{71}$ is 8 with remainder of 7
$8\sqrt{8}$ is 1 with remainder of 0
$8\sqrt{1}$ is 0 with remainder of 1 The remainders backwards are your number in base 8: 10765

The **hexadecimal system, base sixteen**, needs extra symbols to represent the numbers between 10 and 16. The letters A, B, C, D, E, F are commonly used. Hexadecimal is very useful as shorthand notation for base 2. For example, consider the number 0001001110111010 in base 2. Take the number and partition it into 4-digit parts. If these parts are written in base 16 symbols, that is the base 16 number. In other words, $0001001110111010_2 = 13BA_{16}$.

0001	0011	1011	1010					
|	|	|	|					
1	3	B	A					
|	|	|	|					
$1\cdot16^3$	$3\cdot16^2$	$B\cdot16^1$	$A\cdot16^0$	=	$1\cdot4096 + 3\cdot256 + 11\cdot16 + 10$	=	$4096 + 768 + 176 + 10$	= 5050_{10}

Negative numbers

Negative numbers are typically stored in what is known as **two's complement** (and sometimes in one's complement). A number x: $-2^{n-1} \le x \le 2^{n-1} - 1$, can be stored using n bits. Negative numbers are represented as $2^{n-1} - |x|$. The leftmost bit is the sign bit: 0 for positive; 1 for negative

For example, let n = 4 bits. Then $-2^3 \le x \le 2^3-1$ or $-8 \le x \le 7$.

x	base 2	x	$2^3-\|x\|$	base 2
0	0000	-1	8−1=7	1111
1	0001	-2	8−2=6	1110
2	0010	-3	8−3=5	1101
3	0011	-4	8−4=4	1100
4	0100	-5	8−5=3	1011
5	0101	-6	8−6=2	1010
6	0110	-7	8−7=1	1001
7	0111	-8	8−8=0	1000

An advantage to this representation is that you can just add for both addition and subtraction:

5	0101
-2	1110
3	0011

One's complement is found by flipping the bits: 0 becomes 1, 1 becomes 0 . A number between $-2^{n-1}-1 \le x \le 2^{n-1}-1$ can be represented.

x	base 2	x	base 2
1	0001	-1	1110
2	0010	-2	1101
3	0011	-3	1100
4	0100	-4	1011
5	0101	-5	1010
6	0110	-6	1001
7	0111	-7	1000

To add, use binary addition;
add any carry bit, for example:

5	0101	0010
-2	1101	1
3	1|0010	0011

carry bit

A problem with one's complement is that zero has two different representations: 0000 and 1111