CS314: Modular Arithmetic and Tens Complement

A. Demers

September 2003

For those of you who are not familiar with modular arithmetic or are confused by twos complement notation, here are some examples.

Modular Arithmetic. Let M be a positive integer. For any ingeter x, we define $(x \bmod M)$ to be the (nonnegative) remainder of x on division by M. That is, $(x \bmod M)$ is the number r such that

$$0 <= r < M \qquad \text{and} \qquad (\exists q)(x = q \cdot M + r)$$

Here q, the *quotient*, is required to be an integer.

You should convince yourself that $(x \bmod M)$ is *well-defined* – that is, for any x and M there is *exactly* one r satisfying the definition above. This follows from the requirement that r must be in the range $[0..M-1]$. Two distinct values r and r' such that $x = q \cdot M + r)$ and $x = q' \cdot M + r')$ would have to differ by an integer multiple of M (specifically, by $(q' - q) \cdot M$), so r and r' could not both be in the required range.

Suppose a positive number x is represented in some base b. Then $(x \bmod b^k)$ has a particularly simple form – it is just the low order k digits of the representation. Thus, in decimal

$$\begin{aligned}(365 \bmod 10) &= 5 \\ (365 \bmod 100) &= 65\end{aligned}$$

and in binary

$$\begin{aligned}(01101 \bmod 2) &= 1 \\ (01101 \bmod 4) &= 01 = 1 \\ (01101 \bmod 8) &= 101\end{aligned}$$

and so forth.

Addition and subtraction can be consistently performed mod$M$. That is, for any integers x, y and any positive M the two identities

$$(x + y) \bmod M \qquad = \qquad ((x \bmod M) + (y \bmod M)) \bmod M$$

and

$$(x - y) \bmod M \qquad = \qquad ((x \bmod M) - (y \bmod M)) \bmod M$$

can be proved from the definition of the mod function. These identities hold even if x and y are allowed to be negative.

Tens Complement Representation. The twos complement construction we described in lecture works in any base. Here we work through an example in decimal notation, where it is called "tens complement."

Consider numbers represented by two decimal digits. Two decimal digits can represent 100 distinct values, and we usually interpret them as the unsigned values 0 through 100. Instead, we can use the two digits to represent *signed* values, giving us the ability to represent, say, 50 nonnegative numbers and 50 negative numbers (since there are exactly 100 different 2-digit combinations, we can represent 100 different positive and negative numbers altogether). We cleverly choose to represent a number x between -50 and $+49$ using the (unsigned) value (x mod 100). This gives us the following table

$$\begin{array}{rcllcl}
-50 & \mapsto & (-50 \bmod 100) & = & 50 \\
-49 & \mapsto & (-49 \bmod 100) & = & 51 \\
-48 & \mapsto & (-48 \bmod 100) & = & 52 \\
 & & \ldots & & \\
-2 & \mapsto & (-2 \bmod 100) & = & 98 \\
-1 & \mapsto & (-1 \bmod 100) & = & 99 \\
0 & \mapsto & (0 \bmod 100) & = & 0 \\
1 & \mapsto & (1 \bmod 100) & = & 1 \\
2 & \mapsto & (2 \bmod 100) & = & 2 \\
 & & \ldots & & \\
48 & \mapsto & (48 \bmod 100) & = & 48 \\
49 & \mapsto & (49 \bmod 100) & = & 49
\end{array}$$

As a consequence of the identities given above, you can do arithmetic mod100 using the tens complement representations (the rightmost column of the table) and everything "just works." A few examples:

$$\begin{array}{rcl}
1 + 2 = 3 & \text{becomes} & ((1 + 2) \bmod 100) = 3 \\
2 + (-1) = 1 & \text{becomes} & ((2 + 99) \bmod 100) = (101 \bmod 100) = 1 \\
(-49) + 1 = (-48) & \text{becomes} & ((51 + 1) \bmod 100) = 52 \\
(-48) + (-1) = (-49) & \text{becomes} & ((52 + 99) \bmod 100) = (151 \bmod 100) = 51
\end{array}$$

Here's another thing to notice. For every x between (-49) and 49, the representation of $(-x)$ is 100 - (the representation of x). You can prove this statement (generalized to any base) using the definition of the $\bmod$ function; but it should be obvious from looking at the table.

Now, how do you subtract a 2-digit number from 100? Consider the trivial identity

$$(100 - x) \quad = \quad ((99 + 1) - x) \quad = \quad ((99 - x) + 1$$

It is easy to subtract a 2-digit number from 99 (or a 3-digit number from 999, or ...), because the columns behave independently. In each column, you are subtracting a digit value from 9, the largest possible digit value. Thus, it is never necessary to "borrow" from the adjacent column. As we shall see later in this course, the ability to perform multiple operations in parallel without interactions (in this case, to subtract all the digits in parallel simultaneously) is an important factor in building fast hardware.

Now, consider the procedure we have just developed for negating a number x in 2-digit tens complement notation:

> subtract x from 99 and add 1 to the result

and compare it to the procedure for negating a 4-bit number in twos complement discussed in lecture:

> flip the bits of x (equivalent to subtracting x from 1111) and add 1 to the result.

To Think About: In twos complement, you test the sign of a number by looking at the leftmost bit. How do you determine the sign of a number in tens complement? In lecture, we showed how to increase the width of a twos complement binary number (e.g. from 8 to 16 or 32 bits) by "sign extension" – copying the sign bit to the left. What is the analogous procedure in tens complement?