Copyright Rational Software 2002

Book Review

Hacker's Delight
by Henry S. Warren, Jr.

Addison Wesley, 2002
ISBN: 0-201-91465-4
Cover Price: US$39.99
306 Pages

Hacker's Delight by Henry S. Warren Jr. is a delight indeed. Despite its title, this volume of mathematical programming tricks has nothing to do with breaking into other people's computers. Warren is a hacker in the original sense of the word: someone who enjoys writing clever code and finding elegant solutions to computing problems. He discovered many of the tricks and methods he presents in the book during a forty-year career working on compilers and computer architecture with IBM's Research Division. His collection of code snippets and equations range from basic arithmetic operations to sophisticated mathematics.

Computer math may not be a subject that appeals to a broad range of readers, but this book is a gold mine of useful methods and information for those who are interested. The engineer writing a compiler, a math library, or any highly optimized code will find the book indispensable. Anyone familiar with assembly language will find the book accessible and interesting. It can be read from front to back, used as a reference, or enjoyed in the manner Guy Steele describes in his foreword to the book: "Devouring these little programming nuggets was like eating peanuts, or rather bonbons -- I just couldn't stop -- and there was a certain richness to them, a certain intellectual depth, elegance, even poetry."

The book begins by giving a set of formulas for simple operations that one might encounter in the course of writing assembly-level programs. For example, the fastest way to determine whether an unsigned integer is a power of 2 is:

```
x & (x-1)
```

If x is a power of two, the expression evaluates to zero; for other numbers it is nonzero. This section of the book can take days to finish because, for the reader who enjoys recreational bit-wrangling, it is tempting to take a pencil and test each of Warren's assertions. Like many of Warren's formulas, x & (x-1) is not an obvious solution at first glance, but once you

work through it and look at the bits, you can see that it works. In this instance, I tried 8 (a power of 2) and 6 (not a power of 2):

```
x =                     8                           6
x =               0000 1000      or     0000 0110
x-1 =             0000 0111             0000 0101
x & (x-1) =       0000 0000             0000 0100
                       zero                  nonzero
```

The book goes on to give formulas for counting the 1-bits in a word, isolating the rightmost 1-bit, flipping the rightmost contiguous string of 1-bits, shifting and propagating bits, reversing the order of bits and bytes, transposing a matrix, and so on. More than half the book is a collection of methods for doing basic calculations efficiently. Code examples are given either in C or in a simplified assembly language the book defines for 32-bit RISC machines.

Code for Simple to Complex Mathematical Operations

In the course of everyday software engineering, most of us just link our programs with a standard math library and take it for granted that the machine knows how to perform multiplication, division, and other operations on unsigned and signed integers, long integers, floating-point numbers, and so on. This book goes into greater detail, explaining exactly how to do each of these calculations.

After disposing of basic arithmetic operations, Warren turns his attention to more complex math problems -- calculating square roots, for example. His discussion of the subject is both complex and simple. First, he explains Newton's method of computing square roots through a page full of equations that require some effort to follow -- but then he gives an implementation that requires fewer than twenty lines of C code. This is followed by another method that is longer and more cryptic but executes faster, by using a binary search algorithm. Whether you are interested in the equations or merely need the C code to do your job, Warren's solutions are efficient and elegant. Here is his code for computing an integer square root using Newton's method:

```
int isqrt (unsigned x) {
  unsigned x1;
  int s, g0, g1;

  if (x <= 1) return x;
  s = 1;
  x1 = x - 1;
  if (x1 > 65535) {s = s + 8; x1 = x1 >> 16; }
  if (x1 > 255)   {s = s + 4; x1 = x1 >> 8; }
  if (x1 > 15)    {s = s + 2; x1 = x1 >> 4; }
  if (x1 > 3)     {s = s + 1;}

  g0 = 1 << s;   // g0 = 2**s.
  g1 = (g0 + ( x >> s)) >> 1;   //g1 = (g0 + x/g0)/2.
```

```
    while (g1 < g0) {
       g0 = g1;
       g1 = (g0 + (x/g0)) >> 1;
    }
    return g0;
}
```

The book also offers similar solutions for computing cube roots, exponents, and logarithms.

Pursuit of Fascinating Problems

The last few chapters of the book discuss topics chosen seemingly at random from a range of mathematical subjects the author has found interesting. One of the more fascinating chapters, which is far "outside the box" in relation to today's computers, discusses unusual bases for number systems and considers their possible use in computing. Base -2, for example, is a system in which both positive and negative numbers can be represented without using an explicit sign bit. As in the more familiar binary system, negabinary numbers are represented by 0/1 bits -- but the sign flips in every other digit. So instead of the digits being valued (1, 2, 4, 8, 16...), they are valued (0, 1, -2, 4, -8, 16...). The advantage of negabinary numbers is their simplicity in representing negative numbers, but their downside is that negabinary arithmetic operations such as division are quite complicated. Warren also discusses positive and negative complex base systems. In bases in which digits are valued based on powers of the numbers (-1 + i) or (-1 - i), all numbers both real and imaginary can be represented by 0/1 bits. He briefly touches on base e numbers, and finally considers the question of which base is the most efficient one for computing.

Other topics Warren addresses include Gray codes, the Hilbert curve, and prime numbers. Gray codes are a method of arranging the integers from 1 to N in a list so that each number can be visited exactly once by flipping only one bit at a time. The Hilbert curve is a similar idea expressed geometrically: a single continuous curve which, given a space divided into a grid of squares, touches every square exactly once and does not cross itself. In each case, Warren provides both the mathematical discussion and the code to solve the problem.

The chapter on prime numbers is the most challenging mathematically but also one of the most interesting. It starts with a concise overview of various mathematicians' efforts to devise ways of finding prime numbers. "Like many young students, I once became fascinated with prime numbers, and tried to find a formula for them," explains Warren. Actually, this statement is the key to the whole book. The author is one of those people who periodically become fascinated by some problem and devote themselves to learning more about it and searching for a solution.

The chapter ends not with Warren's usual code sample, but instead with an invitation to continue the search for interesting solutions to the problem. Clearly, the author views this book not as a finished collection, but rather as a snapshot of work in progress. After decades of interest-

driven research, Henry Warren has amassed a collection of studies big enough to fill a book. It is fortunate for the rest of us that he has written one, and I look forward to the next installment.

-**Susan McVey**
Software Quality Engineer
Rational Software

For more information on the products or services discussed in this article, please click here and follow the instructions provided. Thank you!

Copyright Rational Software 2002 | Privacy/Legal Information