Network Working Group   S. Crocker
Request for Comments #70   UCLA
                                                               15 October 70

A Note on Padding

The padding on a message is a string of the form 10*. For Hosts with word lengths 16, 32, 48, etc., bits long, this string is necessarily in the last word received from the Imp. For Hosts with word lengths which are not a multiple of 16 (but which are at least 16 bits long), the 1 bit will be in either the last word or the next to last word. Of course if the 1 bit is in the next to last word, the last word is all zero.

An unpleasant coding task is discovering the bit position of the 1 bit within its word. One obvious technique is to repeatedly test the low-order bit, shifting the word right one bit position if the low-order bit is zero. The following techniques are more pleasant.

Isolating the Low-Order Bit

Let W be a non-zero word, where the word length is n. Then W is of the form

```
x....x10....0
\__ __/\__ __/
   V      V
 n-k-1    k
```

where 0<=k<n

and the x's are arbitrary bits.

Assuming two's complement arithmetic,

```
W-1    =      x....x01....1
              _    _
 -W    =      x....x10....0
  _           _    _
  W    =      x....x01....1
```

By using AND, OR and exclusive OR with various pairs of these quantities, useful new forms are obtained.

For example,

```
            W AND W-1       xx...x00....0
                           \__ __/\__ __/
                              V      V
                            n-k-1    k
```

thus removing the low-order 1 bit;

```
also W AND -W =           0....010....0
                         __ __/__ __/
                            V      V
                          n-k-1    k
```

thus isolating the low-order bit.

Below, we will focus solely on this last result; however, in a particular application it may be advantageous to use a variation.

Determining the Position of an Isolated Bit

The two obvious techniques for finding the bit position of an isolated bit are to shift repetitively with tests, as above, and to use floating normalization hardware. On the PDP-10, in particular, the JFFO instruction is made to order*. On machines with hexadecimal normalization, e.g. IBM 360's and XDS Sigma 7's, the normalization hardware may not be very convenient. A different approach uses division and table look-up.

A word with a single bit on has an unsigned integer value of 2^k for $0<=k<n$. If we choose a p such that mod(2^k,p) is distinct for each $0<=k<n$, we can make a table of length p which gives the correspondence between mod(2^k,p) and k. The remainder of this paper is concerned with the selection of an appropriate divisor p for each word length n.

*Some of the CDC machines have a "population count" instruction which gives the number of bits in a word. Note the 2^k-1 has exactly k bits on.

Example

Let n = 8 and p = 11

Then

$$\mathrm{mod}(2^0, 11) = 1$$
$$\mathrm{mod}(2^1, 11) = 2$$
$$\mathrm{mod}(2^2, 11) = 4$$
$$\mathrm{mod}(2^3, 11) = 8$$
$$\mathrm{mod}(2^4, 11) = 5$$
$$\mathrm{mod}(2^5, 11) = 10$$
$$\mathrm{mod}(2^6, 11) = 9$$
$$\mathrm{mod}(2^7, 11) = 7$$

This yields a table of the form

remainder	bit position
0	--
1	0
2	1
3	--
4	2
5	4
6	--
7	7
8	3
9	6
10	5

Good Divisors

The divisor p should be as small as possible in order to minimize the length of the table. Since the divisor must generate n distinct remainders, the divisor will certainly need to be at least n. A remainder of zero, however, can occur only if the divisor is a power of 2. If the divisor is a small power of 2, say 2^j for $j < n-1$, it will not generate n distinct remainders; if the divisor is a larger power of 2, the correspondence table is either 2^{n-1} or 2^n in length. We can thus rule out zero as a remainder value, so the divisor must be at least one more than the word length. This bound is in fact achieved for some word lengths.

Let R(p) be the number of distinct remainders p generates when divided into successively higher powers of 2. The distinct remainders all occur for the R(p) lowest powers of 2. Only odd p are interesting and the following table gives R(p) for odd p between 1 and 21.

p	R(p)	p	R(p)
1	1	13	12
3	2	15	4
5	4	17	8
7	3	19	18
9	6	21	6
11	10		

This table shows that 7, 15, 17 and 21 are useless divisors because there are smaller divisors which generate a larger number of distinct remainders. If we limit our attention to p such that $p > p' \Rightarrow R(p) > R(p')$, we obtain the following table of useful divisors for $p < 100$.

p	R(p)	p	R(p)
1	1	29	28
3	2	37	36
5	4	53	52
9	6	59	58
11	10	61	60
13	12	67	66
19	18	83	82
25	20		

Notice that 9 and 25 are useful divisors even though they generate only 6 and 20 remainders, respectively.

Determination of R(p)

If p is odd, the remainders

$$\mathrm{mod}(2^0,p)$$
$$\mathrm{mod}(2^1,p)$$
$$\vdots$$

will be between 1 and p-1 inclusive. At some power of 2, say 2^t, there will be a repeated remainder, so that for some k < t, $2^k = 2^t \bmod p$.

Since $2^{t+1} = 2^{k+1} \bmod p$

and $2^{t+2} = 2^{k+2} \bmod p$

$$\vdots$$

etc.

all of the distinct remainders occur for $2^0 \ldots 2^{t-1}$. Therefore, R(p)=t.

Next we show that

$$2^{R(p)} = 1 \bmod p$$

We already know that $2^{R(p)} = 2^k \bmod p$

for some 0<=k<R(p). Let j=R(p)-k so 0<j<=R(p). Then

$$2^{k+j} = 2^k \bmod p$$
$$\text{or} \quad 2^j * 2^k = 2^k \bmod p$$
$$\text{or} \quad (2^j - 1) * 2^k = 0 \bmod p$$

Now p does not divide 2^k because p is odd, so p must divide 2^j-1. Thus

$$2^j - 1 = 0 \bmod p$$
$$2^j = 1 \bmod p$$

Since j is greater than 0 by hypothesis and since ther is no k other than 0 less than R(p) such that

$$2^k = 2^0 \bmod p,$$

we must have j=R(p), or $2^{R(p)} = 1 \bmod p$.

We have thus shown that for odd p, the remainders $mod(2^k,p)$ are unique for k = 0, 1,..., R(p)-1 and then repeat exactly, beginning with

$$2^{R(p)} = 1 \bmod p.$$

We now consider even p. Let

$$p = p' * 2^q,$$

where p' is odd. For k<q, $mod(2^k,p)$ is clearly just 2^k because $2^k<p$.

For k>=q,

$$mod(2^k,p) = 2^q * mod(2^{k-q},p').$$

From this we can see that the sequence of remainders will have an initial segment of 1, 2, ...2^{q-1} of length q, and repeating segments of length R(p’). Therefore, R(p) = q+R(p’). Since we normally expect

$$R(p) \sim p,$$

even p generally will not be useful.

I don’t know of a direct way of choosing a p for a given n, but the previous table was generated from the following Fortran program run under the SEX system at UCLA.

```
0
        CALL IASSGN('OC ',56)
1       FORMAT(I3,I5)
        M=0
        DO 100 K=1,100,2
        K=1
        L=0
20      L=L+1
        N=MOD(2*N,K)
        IF(N.GT.1) GO TO 20
        IF(L.LE.M) GO TO 100
        M=L
        WRITE(56,1)K,L
100     CONTINUE
        STOP
        END
```

Fortran program to computer useful divisors

In the program, K takes on trial values of p, N takes on the values of the successive remainders, L counts up to R(p), and M remembers the previous largest R(p). Execution is quite speedy.

Results from Number Theory

The quantity referred to above as R(p) is usually written $Ord_p 2$ and is read "the order of 2 mod p". The maximum value of $Ord_p 2$ is given by Euler’s phi-function, sometimes called the totient. The totient of a positive integer p is the number of integers less than p which are relatively prime to p. The totient is easy to compute from a representation of p as a product of primes:

$$\text{Let } p = p_1^{n_1} * p_2^{n_2} \ldots p_k^{n_k}$$

where the p_i are distinct primes. Then

$$\phi(p) = (p_1 - 1) * p_1^{k_1 - 1} * (p_2 - 1) * p_2^{k_2 - 1} \ldots (p_k - 1) * p_k^{k_k - 1}$$

If p is prime, the totient of p is simply

$$\phi(p) = p-1.$$

If p is not prime, the totient is smaller.

If a is relatively prime to p, then Euler’s generalization of Fermat’s theorem states

$$a^{\phi(m)} = 1 \bmod p.$$

It is this theorem which places an upper bound $Ord_p 2$, because $Ord_p 2$ is the smallest value such that

$$2^{Ord_p 2} = 1 \bmod p$$

Moreover it is always true that phi(p) is divisible by $Ord_p 2$.

Acknowledgements

Bob Kahn read an early draft and made many comments which improved the exposition. Alex Hurwitz assured me that a search technique is necessary to compute R(p), and supplied the names for the quantities and theorems I uncovered.

[This RFC was put into machine readable form for entry]
[into the online RFC archives by Guillaume Lahaye and]
[John Hewes 6/97]