Subject: Re: counting bits
Posted by JD Smith on Tue, 18 Feb 2003 19:56:50 GMT
View Forum Message <> Reply to Message

On Mon, 17 Feb 2003 21:45:42 -0700, Craig Markwardt wrote:

```
>  JD Smith <jdsmith@as.arizona.edu> writes:
>>
>>  For a 2048x2048 array of random long integers, this is 4-6 times as
>>  fast as a more traditional shift method, like the one David suggests.
>>  And here's an ultra-bizzarre one, which requires no look-up table, but
>>  holds its own against the LUT method (note the original array is
>>  destroyed):
>>
>>  arr = ishft(arr AND 'AAAAAAAA'XUL,-1)  + (arr AND '55555555'XUL) arr =
>>  ishft(arr AND 'CCCCCCCC'XUL,-2)  + (arr AND '33333333'XUL) arr =
>>  ishft(arr AND 'F0F0F0F0'XUL,-4)  + (arr AND '0F0F0F0F'XUL) arr =
>>  ishft(arr AND 'FF00FF00'XUL,-8)  + (arr AND '00FF00FF'XUL) arr =
>>  ishft(arr AND 'FFFF0000'XUL,-16) + (arr AND '0000FFFF'XUL)
>>  tot=ulong(total(arr))
>>
>>  See if you can figure that one out ;).
>
>  Very wicked!
>
>  Okay, here's a problem I've always solved by the brute force method:
>
>   * what is the lowest / highest bit position set in an integer?
>
>  For example, a the lowest bit position in the binary number 11010100 is
>  2 (bit positions are labeled starting with 0 of course).  The brute
>  force method involves testing each bit in succession using something
>  like (VALUE AND 2L^I) for each I, until a set bit is found.
>
```

Here's a reasonably fast implementation of your proposed method for
arrays of unsigned longs, to count the highest bit (or rather, the
number of leading 0 bits).

```
function leading_zeroes_reg,num
  num=[num]
   zeroes=make_array(/BYTE,DIMENSION=size(num,/DIMENSIONS),VALU E=255b)
  for i=0,31 do begin
     shft=ishft(num,-(31-i)) AND 1
     zeroes=(zeroes ne 255b)*zeroes+(zeroes eq 255b)* $
            ((shft eq 1)*i+(shft ne 1)*255b)
  endfor
  return, (zeroes eq 255b)*32+(zeroes ne 255b)*zeroes
end
```

And here's an even stranger magical incantation than for bit counting which also does the job:

```
function leading_zeroes,num
  table = [0, 31, 9, 30, 3, 8, 18, 29, 2, 5, 7, 14, 12, 17, $
           22, 28, 1, 10, 4, 19, 6, 15, 13, 23, 11, 20, 16, $
           24, 21, 25, 26, 27]
  c = '7dcd629'XUL
  num= num OR ishft(num,-1)
  num= num OR ishft(num,-2)
  num= num OR ishft(num,-4)
  num= num OR ishft(num,-8)
  num= num OR ishft(num,-16)
  return, (num eq 0)*32+(num ne 0)*table[ishft(c + (c * num),-27)]
end
```

It's about 14 times faster than the pedantic approach, and destroys the array.

What's scary is that somebody must sit around coming up with this stuff. I leave as an exercise the logical inversion required to calculate trailing zeroes.

JD

P.S. Don't try these on signed integers.