Subject: Re: counting bits
Posted by JD Smith on Mon, 17 Feb 2003 23:54:23 GMT
View Forum Message <> Reply to Message

On Mon, 17 Feb 2003 13:41:20 -0700, David Fanning wrote:

> David Fanning (david@dfanning.com) writes:
>
>> IDL> for j=0,31 do total = TOTAL((array AND 2^j) NE 0)
>
> Whoops, better make that "2" a long integer:
>
> IDL> for j=0,31 do total = TOTAL((array AND 2L^j) NE 0)
>

There are lots of efficient algorithms in C for counting bits. This isn't one of them ;). The least efficient which resembles this would be shift-and-add, ala:

```
for j=0,31 do begin
  tot=tot+(arr AND 1L)
  arr=ishft(temporary(arr),-1)
endfor
tot=ulong(total(tot))
```

This is also quite slow (even slower than David's, in fact, which suggests IDL's ISHFT is pretty slow). The method I found fastest (among those I've tried), is a pretty silly and straightforward one, namely table lookup:

```
bits = [0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4, $ ; 0   - 15  */
        1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5, $ ; 16  - 31  */
        1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5, $ ; 32  - 47  */
        2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6, $ ; 48  - 63  */
        1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5, $ ; 64  - 79  */
        2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6, $ ; 80  - 95  */
        2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6, $ ; 96  - 111 */
        3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7, $ ; 112 - 127 */
        1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5, $ ; 128 - 143 */
        2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6, $ ; 144 - 159 */
        2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6, $ ; 160 - 175 */
        3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7, $ ; 176 - 191 */
        2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6, $ ; 192 - 207 */
        3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7, $ ; 208 - 223 */
        3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7, $ ; 224 - 239 */
        4, 5, 5, 6, 5, 6, 6, 7, 5, 6, 6, 7, 6, 7, 7, 8  ] ; 240 - 255 */

tot=ulong(total(bits[rand_arr AND 'FF'XUL] + $
```

```
        bits[ishft(rand_arr,-8)  AND 'FF'XUL]+ $
        bits[ishft(rand_arr,-16) AND 'FF'XUL]+ $
        bits[ishft(rand_arr,-24) AND 'FF'XUL]))
```

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 ;).

Good luck,

JD
