Subject: Re: counting bits
Posted by David Fanning on Mon, 17 Feb 2003 20:38:24 GMT
View Forum Message <> Reply to Message

Joe Foose (jfoose@ameritech.net) writes:

> In IDL, does anybody have any ideas of how to efficiently count all the bits
> that are set in an array of unsigned long integers? I'm not concerened with
> which bits are set, but just how many. thanks, joe.

Just of the top of my head, wouldn't something like this work:

```
IDL> total = 0.0
IDL> for j=0,31 do total = TOTAL((array AND 2^j) NE 0)
IDL> print, Fix(total)
```

Cheers,

David
--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: counting bits
Posted by David Fanning on Mon, 17 Feb 2003 20:41:20 GMT
View Forum Message <> Reply to Message

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)
```

Cheers,

David
--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

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

Subject: Re: counting bits
Posted by Craig Markwardt on Tue, 18 Feb 2003 04:45:42 GMT
View Forum Message <> Reply to Message

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.

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: counting bits
Posted by David Fanning on Tue, 18 Feb 2003 05:09:59 GMT
View Forum Message <> Reply to Message

Craig Markwardt (craigmnet@cow.physics.wisc.edu) writes:

> 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.

You two guys really have to get a life. Are either
of you two married?

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: counting bits
Posted by eddie haskell on Tue, 18 Feb 2003 16:10:29 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:

> 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.

I have not done any checking to see if this is any faster than a brute force method but you could use something like this to get the lowest and highest set bits:

```
IDL> n = 212   ;11010100
IDL> nbits = 8
IDL> print,(indgen(nbits))[(where((n and 2l^lindgen(nbits)) ne 0))[[0,nbits]]]

      2       7
```

An issue with this is that N=0 will return [0,0] (the same as N=1), but checking for a N of zero can easily be done beforehand.

Cheers,
eddie

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.

Subject: Re: counting bits
Posted by Rick Towler on Tue, 18 Feb 2003 21:27:23 GMT
View Forum Message <> Reply to Message

"David Fanning" wrote

> Craig Markwardt writes:
>
>> 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?
>>
>
> You two guys really have to get a life.

Now David, isn't this the pot calling the kettle black? With 4,110 posts to this newsgroup some would say you need to get a life! :)

-Rick

Subject: Re: counting bits
Posted by David Fanning on Tue, 18 Feb 2003 21:55:22 GMT
View Forum Message <> Reply to Message

Rick Towler (rtowler@u.washington.edu) writes:

> Now David, isn't this the pot calling the kettle black? With 4,110 posts to
> this newsgroup some would say you need to get a life! :)

I don't know. Anyone who is counting my posts is a

little suspect, too. :-)

No, I just have this sister-in-law who is well,
you know, nice. And it just occurred to me that
JD and Craig may have a little too much free time on
their hands, if you know what I mean. I thought I'd
try to kill two birds with the same stone.

Cheers,

David

P.S. Let's just say people from my wife's family aren't
really too picky, and they *really* appreciate strong
mathematical types. :-)

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: counting bits
Posted by JD Smith on Tue, 18 Feb 2003 22:58:27 GMT
View Forum Message <> Reply to Message

On Tue, 18 Feb 2003 14:55:22 -0700, David Fanning wrote:

> Rick Towler (rtowler@u.washington.edu) writes:
>
>> Now David, isn't this the pot calling the kettle black? With 4,110
>> posts to this newsgroup some would say you need to get a life! :)
>
> I don't know. Anyone who is counting my posts is a little suspect, too.
> :-)
>
> No, I just have this sister-in-law who is well, you know, nice. And it
> just occurred to me that JD and Craig may have a little too much free
> time on their hands, if you know what I mean. I thought I'd try to kill
> two birds with the same stone.
>

I hope your sister-in-law (or wife, for that matter) isn't a user of
Google Groups. Also, I'm flattered you'd want me in your family, but
my wife would probably frown on it. And to defend myself, I collected
all those bit fiddling tricks while writing a 64-bit endgame solver

for Reversi years ago. Hmm, I guess that isn't exactly too compelling
of a defense against the "Get a life" rubric, but it will have to
suffice.

JD

Subject: Re: counting bits
Posted by Craig Markwardt on Tue, 18 Feb 2003 23:10:38 GMT
View Forum Message <> Reply to Message

"Rick Towler" <rtowler@u.washington.edu> writes:

> "David Fanning" wrote
>
>> Craig Markwardt writes:
>>
>>> 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?
>>>
>>
>> You two guys really have to get a life.
>
> Now David, isn't this the pot calling the kettle black? With 4,110 posts to
> this newsgroup some would say you need to get a life! :)

Yeah, my excuse is that I've been snowed in for three days, what's
yours, Dr. David "Matchmaker" Franning?

--
 -- --------------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------

Subject: Re: counting bits
Posted by Dick Jackson on Wed, 19 Feb 2003 15:47:31 GMT
View Forum Message <> Reply to Message

[sorry for the late entry, it seems my news server has been failing when
I attach a file to a posting :-(]

"JD Smith" <jdsmith@as.arizona.edu> wrote in message
news:pan.2003.02.17.23.54.23.693563.14101@as.arizona.edu...

> The method I found fastest
> (among those I've tried), is a pretty silly and straightforward one,
> namely table lookup:

I don't think it's silly at all, but taking it one step further will
really speed it up. If you have enough extra memory for it, just convert
the whole ULong array to bytes, do one lookup and you're done. One more
'gotcha' came up when running this:

Data:

```
rand_arr = ULIndgen(2048, 2048)
```

JD's method:

```
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]))
```

Dick's method:

```
byte_rand_arr = Byte(rand_arr, 0, N_Elements(rand_arr)*4)
tot = ULong(Total(bits[byte_rand_arr]))
```

When I tried this, I got:

```
IDL> CountingBits
IShft-AND-lookup method:      2.063 seconds.
tot =     46137328
Byte-lookup method:      0.691 seconds.
tot =     46137292
```

Uh-oh... I set the Total calls to have /Double and then we both get the
same (I hope correct) answer:

```
IDL> CountingBits
IShft-AND-lookup method:      2.063 seconds.
tot =     46137344
Byte-lookup method:      0.671 seconds.
tot =     46137344
```

My kingdom for a /Long flag on Total()!

I attach my test program with handy bonus timer routines TStart and
TReport. [Actually copied in-line below, now]

Now, if there were a way to convert to Byte without copying...

Cheers,
--
-Dick

Dick Jackson / dick@d-jackson.com
D-Jackson Software Consulting / http://www.d-jackson.com
Calgary, Alberta, Canada / +1-403-242-7398 / Fax: 241-7392

===== CountingBits.pro =====

```
PRO TStart, msg                 ; Timer Start
                                ; Save current time for use by
TReport
COMMON Timer, t0
IF N_Elements(msg) NE 0 THEN Print, msg
t0 = SysTime(1)
END

PRO TReport, msg                ; Timer Report
                                ; Print elapsed time since last
TStart
COMMON Timer, t0
IF N_Elements(msg) EQ 0 THEN msg = ''
Print, Format='(A0, D10.3," seconds.")', msg, SysTime(1)-t0
END

PRO CountingBits

rand_arr = ULIndgen(2048, 2048)

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 */
```

```
TStart

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], /Double))

TReport, 'IShft-AND-lookup method: '

Print, 'tot = ', tot

TStart

byte_rand_arr = Byte(rand_arr, 0, N_Elements(rand_arr)*4)
tot = ULong(Total(bits[byte_rand_arr], /Double))

TReport, 'Byte-lookup method: '

Print, 'tot = ', tot

END
```

Subject: Re: counting bits
Posted by Craig Markwardt on Thu, 20 Feb 2003 05:31:32 GMT
View Forum Message <> Reply to Message

JD Smith <jdsmith@as.arizona.edu> writes:

```
>
>  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
```

Thanks. Thinking about it further, one could probably do a pretty fast look up table on a byte-by-byte basis, similar to the counting-bits lookup that you presented initially.

Craig

--
 -- -------------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- -------------

Subject: Re: counting bits
Posted by JD Smith on Thu, 20 Feb 2003 15:24:26 GMT
View Forum Message <> Reply to Message

On Wed, 19 Feb 2003 22:31:32 -0700, Craig Markwardt wrote:

> JD Smith <jdsmith@as.arizona.edu> writes:
>>
>> 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
>
> Thanks. Thinking about it further, one could probably do a pretty fast
> look up table on a byte-by-byte basis, similar to the counting-bits
> lookup that you presented initially.
>

Yes, but it wouldn't have the street cred of '7dcd629'XUL.

JD

Subject: Re: counting bits
Posted by JD Smith on Thu, 20 Feb 2003 15:43:26 GMT

On Wed, 19 Feb 2003 08:47:31 -0700, Dick Jackson wrote:

> [sorry for the late entry, it seems my news server has been failing when
> I attach a file to a posting :-(]
>
> "JD Smith" <jdsmith@as.arizona.edu> wrote in message
> news:pan.2003.02.17.23.54.23.693563.14101@as.arizona.edu...
>
>> The method I found fastest
>> (among those I've tried), is a pretty silly and straightforward one,
>> namely table lookup:
>
> I don't think it's silly at all, but taking it one step further will
> really speed it up. If you have enough extra memory for it, just convert
> the whole ULong array to bytes, do one lookup and you're done. One more
> 'gotcha' came up when running this:
>

Good idea, and fast indeed.

> Data:
>
> rand_arr = ULIndgen(2048, 2048)
>
> JD's method:
>
> 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]))
>
> Dick's method:
>
> byte_rand_arr = Byte(rand_arr, 0, N_Elements(rand_arr)*4) tot =
> ULong(Total(bits[byte_rand_arr]))
>
> When I tried this, I got:
>
> IDL> CountingBits
> IShft-AND-lookup method: 2.063 seconds. tot = 46137328
> Byte-lookup method: 0.691 seconds. tot = 46137292
>
> Uh-oh... I set the Total calls to have /Double and then we both get the
> same (I hope correct) answer:
>
> IDL> CountingBits
> IShft-AND-lookup method: 2.063 seconds. tot = 46137344

> Byte-lookup method: 0.671 seconds. tot = 46137344

That's very strange. Here's what I get for my four independent
methods without any /DOUBLE:

```
     3.4187140
  46137344
     6.9928349
  46137344
     1.2564960
  46137344
     1.2767580
  46137344
```

Indeed:

```
IDL> print,ulong(total(bits[rand_arr AND 'FF'XUL] + $
IDL>                bits[ishft(rand_arr,-8)  AND 'FF'XUL]+ $
IDL>                bits[ishft(rand_arr,-16) AND 'FF'XUL]+ $
IDL>                bits[ishft(rand_arr,-24) AND 'FF'XUL], /Double))
   46137344
IDL> print,ulong(total(bits[rand_arr AND 'FF'XUL] + $
IDL>                bits[ishft(rand_arr,-8)  AND 'FF'XUL]+ $
IDL>                bits[ishft(rand_arr,-16) AND 'FF'XUL]+ $
IDL>                bits[ishft(rand_arr,-24) AND 'FF'XUL]))
   46137344
```

I'm not sure what the difference could be (other than Win vs. Linux).

One thing I did notice when creating "random" arrays:

```
IDL> print,FORMAT='(F5.2,A)',total(ulong(randomu(sd,100)*2.^31) mod 2 eq 1),$
         '% odd'
```

Try this a few times. That lowest bit just does not get set. Some
floating-point representation expert must have an explanation.

> My kingdom for a /Long flag on Total()!

I'll throw mine in also.

JD
