Subject: 32-bit Unsigned Integers, Was: Unsigned Integers - How?
Posted by davidf on Sat, 08 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Peter Berdeklis <peter@atmosp.physics.utoronto.ca> carries on this discussion about 16-bit unsigned integers when he writes:

> Unfortunately, I'm not reading 16-bit integers but 32-bit integers.
> Sorry I forgot to mention that. So how would I pull the same trick
> with 32-bit integers?

Somehow, Peter, I just *knew* you didn't have 16-bit integers! :-)

Alright, I know how to answer part of this question. Perhaps we can get some help with the rest.

Suppose you have 100 32-bit *unsigned* integers in a data file. Read them into IDL long *signed* integers. Like this:

```
data = LONARR(100)
READU, lun, datafile, data
```

First of all, if your data values are all less than 2L^31-1 or 2147483647 you are home free, don't worry about a thing. If your data has values greater than that, things get a little dicey. (Note that MAX(data) won't help much here because values of 2L^31 and higher will show as *negative* values. You basically will have to know this some other way.)

Now, here is where I start to get unsure of myself. I know how to turn *one* unsigned 32-bit integer into its real value. You use the BYTE function to individually read the four bytes of information in the 32-bit integer and you reconstruct those bytes into a DOUBLE-PRECISION value. The code looks like this:

```
number = data(0)
factor = 256.0D
realNumber = BYTE(number, 0)*factor^3 + BYTE(number,1)*factor^2 +$
  BYTE(number,2)*factor^1 + BYTE(number,3)*factor^0
```

This is for a big endian machine, like most UNIX machines. If you are on a little endian machine (like a PC), you will have to reverse the order in which the real number is constructed. Your code will look like this:

```
number = data(0)
factor = 256.0D
```

```
  realNumber = BYTE(number, 0)*factor^0 + BYTE(number,1)*factor^1 +$
    BYTE(number,2)*factor^2 + BYTE(number,3)*factor^3
```

What I don't know how to do (perhaps Bill Thompson or Mitchell Grunes
can help us here), is how to do this for the whole array at once in
an "array" type way. I certainly know how to write a loop! :-)

```
  realNumbers = DBLARR(N_ELEMENTS(data))
  factor = 256.0D
  FOR j=0, N_ELMENTS(data)-1 DO BEGIN
     realNumbers(j) = BYTE(data(j), 0)*factor^0 + BYTE(data(j),1)*factor^1 +$
     BYTE(data(j),2)*factor^2 + BYTE(data(j),3)*factor^3
  ENDFOR
```

Perhaps that will get us started.

> By the way David, I just looked up your Web page. Thanks for the tips.
> Since you worked at RSI, do you know why IDL doesn't have an
> unsigned data type?

No, I don't know why. Perhaps they believe idle minds are the
devil's workshop. :-)

Goin' dancin'. See you later...

David

David Fanning, Ph.D.
Fanning Software Consulting
2642 Bradbury Court, Fort Collins, CO 80521
Phone: 970-221-0438 Fax: 970-221-4762
E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com

Subject: Re: 32-bit Unsigned Integers, Was: Unsigned Integers - How?
Posted by Peter Berdeklis on Mon, 10 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

On Sat, 8 Feb 1997, David Fanning wrote:
> Somehow, Peter, I just *knew* you didn't have 16-bit integers! :-)

Hard to believe that's cutting edge on a PC, isn't it? :-}

Thank you for your suggestion. It works perfectly. In return, here's
how to do it for an array.

```
th = '8f7b91dd'x  ; Unsigned this is 2407240157.
    ; Signed this is -1887727139.

th_arr = replicate(th, 10) ; make an array of 10 of them to test

tb_arr = byte(th_arr, 0, 4, 10) ; map the array to a 4x10 array of bytes
    ;  if the initial array is not 1D just add
    ;  dimensions (up to 6 more)

; make your array of 10 doubles

td_arr = double(tb_arr(0,*)) + double(tb_arr(1,*)*2.^8) $
          + double(tb_arr(2,*)*2.^16) + double(tb_arr(3,*)*2.^24)
```

Of course, as you said, the byte order is dependent on the endian of the machine. However, there is no need to check if the unsigned has overflowed since this will correctly map values that haven't. Of course if you do want to check, just check if the number is <0. It shouldn't be if it is unsigned. :)

Thanks again for the help.

Peter Berdeklis
Dept. of Physics, Univ. of Toronto

Subject: Re: 32-bit Unsigned Integers, Was: Unsigned Integers - How?
Posted by James Tappin on Tue, 11 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

It has come to my attention that in my post explaining how to handle unsigned 32-bit integer arrays I allowed some artefacts to enter the displayed code: namely the string "$<3>" at the end of some lines. This is not a part of the line it is just the result of a bug in IDL's line-editing options. (In fact it's only in the two print lines which are only there to illustrate what is happening).

--
 +-----------------------+-------------------------------- --+---------+
James Tappin,	School of Physics & Space Research	O__
sjt@star.sr.bham.ac.uk	University of Birmingham	-- \/`
Ph: 0121-414-6462. Fax: 0121-414-3722	
 +-- --+---------+

Subject: Re: 32-bit Unsigned Integers, Was: Unsigned Integers - How?
Posted by James Tappin on Tue, 11 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
[SNIP]

```
>
>  Now, here is where I start to get unsure of myself. I know
>  how to turn *one* unsigned 32-bit integer into its real value.
>  You use the BYTE function to individually read the four
>  bytes of information in the 32-bit integer and you reconstruct
>  those bytes into a DOUBLE-PRECISION value. The code looks
>  like this:
>
>     number = data(0)
>     factor = 256.0D
>     realNumber = BYTE(number, 0)*factor^3 + BYTE(number,1)*factor^2 +$
>        BYTE(number,2)*factor^1 + BYTE(number,3)*factor^0
>
>  This is for a big endian machine, like most UNIX machines. If
>  you are on a little endian machine (like a PC), you will have to
>  reverse the order in which the real number is constructed. Your
>  code will look like this:
>
>     number = data(0)
>     factor = 256.0D
>     realNumber = BYTE(number, 0)*factor^0 + BYTE(number,1)*factor^1 +$
>        BYTE(number,2)*factor^2 + BYTE(number,3)*factor^3
>
>  What I don't know how to do (perhaps Bill Thompson or Mitchell Grunes
>  can help us here), is how to do this for the whole array at once in
>  an "array" type way. I certainly know how to write a loop! :-)
>
>     realNumbers = DBLARR(N_ELEMENTS(data))
>     factor = 256.0D
>     FOR j=0, N_ELMENTS(data)-1 DO BEGIN
>        realNumbers(j) = BYTE(data(j), 0)*factor^0 + BYTE(data(j),1)*factor^1 +$
>        BYTE(data(j),2)*factor^2 + BYTE(data(j),3)*factor^3
>     ENDFOR
>
```

Here's my version in vector form

```
IDL> a = '7ffffff0'xl
IDL> print,a$<3>
  2147483632
IDL> x = lindgen(32)+a
IDL> print,x$<3>
```

```
  2147483632  2147483633  2147483634  2147483635  2147483636  2147483637
  2147483638  2147483639  2147483640  2147483641  2147483642  2147483643
  2147483644  2147483645  2147483646  2147483647 -2147483648 -2147483647
 -2147483646 -2147483645 -2147483644 -2147483643 -2147483642 -2147483641
 -2147483640 -2147483639 -2147483638 -2147483637 -2147483636 -2147483635
 -2147483634 -2147483633
IDL> bx = byte(x,0,4,n_elements(x))
IDL> factor = 256.0d0^(3-indgen(4))
IDL> xr = total(bx*factor(*,intarr(n_elements(x))),1)
IDL> print,xr
   2.1474836e+09   2.1474836e+09   2.1474836e+09   2.1474836e+09
   2.1474836e+09   2.1474836e+09   2.1474836e+09   2.1474836e+09
   2.1474836e+09   2.1474836e+09   2.1474836e+09   2.1474836e+09
   2.1474836e+09   2.1474836e+09   2.1474836e+09   2.1474836e+09
   2.1474836e+09   2.1474836e+09   2.1474836e+09   2.1474837e+09
   2.1474837e+09   2.1474837e+09   2.1474837e+09   2.1474837e+09
   2.1474837e+09   2.1474837e+09   2.1474837e+09   2.1474837e+09
   2.1474837e+09   2.1474837e+09   2.1474837e+09   2.1474837e+09
```

For a little-endian machine, factor becomes 256.0d0^indgen(4)

--


```
 +----------------------+--------------------------------- --+---------+
| James Tappin,         | School of Physics & Space Research |  O__    |
| sjt@star.sr.bham.ac.uk | University of Birmingham       | --  \/` |
| Ph: 0121-414-6462. Fax: 0121-414-3722         |  |
 +-------------------------------------------------------- --+---------+
```

Subject: Re: 32-bit Unsigned Integers, Was: Unsigned Integers - How?
Posted by plugge on Thu, 13 Feb 1997 08:00:00 GMT

View Forum Message <> Reply to Message

In article <330039AF.23DB@star.sr.bham.ac.uk>, James Tappin
<sjt@star.sr.bham.ac.uk> writes:
|>David Fanning wrote:
|>[SNIP]
|>
|>>
|>> Now, here is where I start to get unsure of myself. I know
|>> how to turn *one* unsigned 32-bit integer into its real value.
|>> You use the BYTE function to individually read the four
|>> bytes of information in the 32-bit integer and you reconstruct
|>> those bytes into a DOUBLE-PRECISION value. The code looks
|>> like this:
|>>
|>>   number = data(0)
|>>   factor = 256.0D

```
|>>    realNumber = BYTE(number, 0)*factor^3 + BYTE(number,1)*factor^2 +$
|>>       BYTE(number,2)*factor^1 + BYTE(number,3)*factor^0
|>>
|>> This is for a big endian machine, like most UNIX machines. If
|>> you are on a little endian machine (like a PC), you will have to
|>> reverse the order in which the real number is constructed. Your
|>> code will look like this:
|>>
|>>    number = data(0)
|>>    factor = 256.0D
|>>    realNumber = BYTE(number, 0)*factor^0 + BYTE(number,1)*factor^1 +$
|>>       BYTE(number,2)*factor^2 + BYTE(number,3)*factor^3
|>>
|>> What I don't know how to do (perhaps Bill Thompson or Mitchell Grunes
|>> can help us here), is how to do this for the whole array at once in
|>> an "array" type way. I certainly know how to write a loop! :-)
|>>
|>>    realNumbers = DBLARR(N_ELEMENTS(data))
|>>    factor = 256.0D
|>>    FOR j=0, N_ELMENTS(data)-1 DO BEGIN
|>>       realNumbers(j) = BYTE(data(j), 0)*factor^0 + BYTE(data(j),1)*factor^1
|>+$
|>>       BYTE(data(j),2)*factor^2 + BYTE(data(j),3)*factor^3
|>>    ENDFOR
|>>
|>
|>Here's my version in vector form
|>
```

another vector version:

```
x=2000000000L+ 1000000000L ;a large integer
help,x
y=replicate(x, 4,4) ;generate the array
y(0,0)=1  ;some values should be positive
y(1,1)=10
y(2,2)=100
y(3,3)=1000
print,y
w=where(y lt 0)  ;check for negative values
s=size(y)
dbl=make_array(/double,size=s) ;make the new array
help,dbl
dbl(0,0)=y
dbl(w)=dbl(w)+4294967296d0 ;that's it!! ;-)))
print,dbl
```