Subject: Re: 10 bytes real
Posted by Craig Markwardt on Thu, 26 Oct 2000 15:31:05 GMT
View Forum Message <> Reply to Message

Thierry Wannier <thierry.wannier@unifr.ch> writes:

> I want to analyse with IDL data obtained in the lab.
> For this, I have to transfer and read on my PC the data which were
> generated by a "home made" software running on a Mac.
> Transfer: OK
> Read with IDL: ....
> I first fought successfully against a tribe a big-endians integer, then
> found a way to cope with Booleans but now "I feel like a motherless
> child" in front of a bunch of 10 bytes reals (IEEE).
>
> Some suggestions?
> Thanks: T.Wannier

Ten-byte reals. Ughh. Are you sure you can use something normal like
8-byte reals? IDL doesn't do 10-byte.

For 4- and 8-byte reals I am personally hooked on using IEEE_TO_HOST
and HOST_TO_IEEE from the IDL Astronomy Library. It has the nifty
benefit converting any standard network-order type, including
integers, into the host-specific endianness. Here is how I use it with
floating point numbers.

```
bb = bytarr(8)      ;; Read ten bytes of data
readu, unit, bb

ff = double(bb, 0)  ;; Cast to double, but still with the wrong byte order

ieee_to_host, ff    ;; Convert to host byte order
```

Craig

P.S. You will have to download at least where_negzero.pro
conv_unix_vax.pro as well.

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

Subject: Re: 10 bytes real

Posted by promashkin on Thu, 26 Oct 2000 22:07:42 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:

> bb = bytarr(8) ;; Read ten bytes of data
> readu, unit, bb

Sorry, Craig, I am not following you here. How exactly does the above
read 10 bytes into an eight-element array?
I guess I've done enough IDLing for one day. Can't figure a one-liner
out :-(

Pavel

---

Subject: Re: 10 bytes real
Posted by Craig Markwardt on Thu, 26 Oct 2000 22:40:07 GMT
View Forum Message <> Reply to Message

Pavel Romashkin <promashkin@cmdl.noaa.gov> writes:
> Craig Markwardt wrote:
>
>> bb = bytarr(8) ;; Read ten bytes of data
>> readu, unit, bb
>
> Sorry, Craig, I am not following you here. How exactly does the above
> read 10 bytes into an eight-element array?
> I guess I've done enough IDLing for one day. Can't figure a one-liner
> out :-(

[Legal eagle mode]

If you look carefully, I only talked about 4- and 8-byte floating
point numbers. I have no idea how to read 10-byte ones. This is an
Intel-ism that IDL doesn't seem to support.

Craig

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

---

Subject: Re: 10 bytes real

Posted by Thierry Wannier on Fri, 27 Oct 2000 06:20:32 GMT
View Forum Message <> Reply to Message

Thanks for the idea, but unfortunately it does not solve my problem, since the data file really contains 10 bytes reals.
I thought that a way to turn around the problem would be to
a) read the ten bytes,
b) reorder them in little-endian (PC type if I recall correctly)
c) read the components of the number (i.e. decompose the real in its significand and exponent parts (that's how I do understand reals are build up))
d) recompose a real (double precision: 16 bytes) using this information.

Unfortunately, I am no computer specialist but just a middle range user, and I have
no idea about the possibilities of doing this decomposition/recomposition of a real number.

T.

---

Subject: Re: 10 bytes real
Posted by Karl Schultz on Fri, 27 Oct 2000 15:39:25 GMT
View Forum Message <> Reply to Message

"Thierry Wannier" <thierry.wannier@unifr.ch> wrote in message news:39F91EB0.7EE5066A@unifr.ch...
> Thanks for the idea, but unfortunately it does not solve my problem, since the
> data file really contains 10 bytes reals.
> I thought that a way to turn around the problem would be to
> a) read the ten bytes,
> b) reorder them in little-endian (PC type if I recall correctly)
> c) read the components of the number (i.e. decompose the real in its
> significand and exponent parts (that's how I do understand reals are build
> up))

This is tricky but you *could* do it...

80-bit IEEE floats use a 65 bit signed mantissa and a 15 bit signed exponent.
64-bit IEEE uses 53 and 11, respectively.
Handling the mantissa is easy - just toss the lower 12 bits.
You'd have to check the exponents before fixing them up because if the magnitude of the 15 bit exponent is so big that it won't fit into 11 bits, you've got a number that is too large, and you end up with an effective overflow. You'd have to watch underflow as well.
I also think that the exponent is stored in "excess" format, meaning that,

for 11-bit exponents, the exponent is stored as [0..2047] instead of
[-1024..1023]. Check the IEEE specs to be sure. But I think you'd have to:
load the 15-bit exponent, subtract 2^14, clamp to [-1024..1023] and then add
1024.

Also, the 80x87 math processors on wintel machines are 80-bit anyway and I
think that there are instructions that would load 80-bit floats into the
floating point regs. After you've done that, you can read them back out as
a double. I don't know if there is any C compiler support. You might be
able to pull some #asm tricks.

Finally, I noticed some hints at 80-bit support for the PowerMac in float.h
that comes with MS C++ for windows. Maybe the Mac C compilers have 80-bit
float support.

> d) recompose a real (double precision: 16 bytes) using this information.
>
> Unfortunately, I am no computer specialist but just a middle range user,
and I
> have
> no idea about the possibilities of doing this decomposition/recomposition
of a
> real number.
>
> T.
>

---

Subject: Re: 10 bytes real
Posted by davidf on Fri, 27 Oct 2000 16:08:42 GMT
View Forum Message <> Reply to Message

Karl Schultz (kschultz@researchsystems.com) writes:

> This is tricky but you *could* do it...

I just remembered the reason I didn't go into computer science. :-(

Cheers,

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

---

Subject: Re: 10 bytes real
Posted by Craig Markwardt on Fri, 27 Oct 2000 16:39:01 GMT
View Forum Message <> Reply to Message

Thierry Wannier <thierry.wannier@unifr.ch> writes:

> Thanks for the idea, but unfortunately it does not solve my problem, since the
> data file really contains 10 bytes reals.
> I thought that a way to turn around the problem would be to
> a) read the ten bytes,
> b) reorder them in little-endian (PC type if I recall correctly)
> c) read the components of the number (i.e. decompose the real in its
> significand and exponent parts (that's how I do understand reals are build
> up))
> d) recompose a real (double precision: 16 bytes) using this information.
>
> Unfortunately, I am no computer specialist but just a middle range user, and I
> have
> no idea about the possibilities of doing this decomposition/recomposition of a
> real number.

If you can do the research and find the specs on Intel 10-bit reals,
we can probably figures something out.

I found the following bit ranges:

| | float | double | 80-bit |
|---|---|---|---|
| sign | 31 | 63 | 79 |
| exponent | 23-30 | 52-62 | 56-78 |
| mantissa | 0-22 | 0-51 | 0-55 |

It seems 10-byte format supports a *really* large exponent. That's
pretty wierd. You have it tough because 10-bytes is not an even
multiple of any of the IDL types. Thus you will need to insert your
10xN array into a 16xN array and then convert to ULONG64. Then the
bit twizzling comes. Here ISHFT and AND will be your friends.

Craig

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

Subject: Re: 10 bytes real

Posted by Ed Santiago on Mon, 30 Oct 2000 13:39:23 GMT
View Forum Message <> Reply to Message

Enclosed below is an IDL function I wrote earlier this year. It parses 16-byte FLOATs, used on some legacy VAX code whose results I needed to work with.

It doesn't do over/underflow checks (I "know" what the data should look like, and so didn't feel like taking the time to do it right).

My code also deals with ULONGS for input. You'll need to change that to UINTs, and change all the bitmasks.

In short, this doesn't solve your problem, but it might be a start.

G'luck,
^E

snip oo oo oo oo oo
---------\---------\---------\---------\---- ---------\

```
;+
; NAME:
;    PARSE_REAL16
;
; IDENT:
;    $Id: parse_real16.pro,v 1.1 2000/04/26 14:51:16 esm Exp $
;
; PURPOSE:
;    Convert (VAX Fortran) REAL16 (16-byte floats) to float or double
;
; AUTHOR:
;    Ed Santiago
;
; CALLING SEQUENCE:
;    float = parse_real16( real16 )
;
; INPUTS:
;    real16      4xn array of ULONGs.
;
; OUTPUTS:
;    float       array of length n, with machine-readable IEEE floats/doubles
;
; KEYWORDS:
;    /DOUBLE     convert to 8-byte (64-bit) double-precision IEEE T_float
;
; SIDE EFFECTS:
;
; EXAMPLE:
```

```
;
; ACKNOWLEDGMENTS:
;     The author wishes to acknowledge Evan Noveroske for patiently
;     describing VAX Fortran and filesystem stuff, figuring out the
;     "record" stuff in VAX files and how to read them in UNIX,
;     generating sample data files, finding documentation on the
;     internal representation of REAL*16, and most especially
;     for his kindness and promptness in providing this help!
;
;-
FUNCTION parse_real16, real16, double=double, to=to

  On_Error, 2

  ; Parse the keywords.
  IF N_Elements(to) EQ 0 THEN to = 4
  IF Keyword_Set(double) THEN to = 8

  IF to NE 4 AND to NE 8 THEN MESSAGE, 'Can only convert to 4 or 8 bytes'

  ; Input argument MUST be a 4xn array of ULONGs.  Anything else, and we die
  IF  size(real16, /TName) NE 'ULONG' THEN $
    MESSAGE, 'Input argument must be of type ULONG'
  IF (size(real16))[1] NE 4 THEN $
    MESSAGE, 'Input argument must be a 4xN array'

  ;
  ; Okay, here we go.
  ;
  ; For more details on binary representations, see
  ;
  ;          http://www.digital.com/fortran/docs/vms-um/dfum020.htm
  ;
  ; REAL16 is a 128-bit quantity, defined as follows:
  ;
  ;   31           24 23          16 15           8 7            0
  ;    +---------------+---------------+---------------+-------- --------+
  ;   |S|         exponent          |          mantissa           |
  ;    +-+-----------------------------+----------------------- --------+
  ;   |                         mantissa                        |
  ;    +------------------------------------------------------- --------+
  ;   |                         mantissa                        |
  ;    +------------------------------------------------------- --------+
  ;   |                         mantissa                        |
  ;    +------------------------------------------------------- --------+
  ;
  ; where:
  ;
```

```
;     SIGN, as always, is the single highest bit.  0 = positive, 1 = neg
;
;     EXPONENT is an "excess 16384" exponent, more on that later.
;
;     MANTISSA is the fractional part, with the "redundant most significant
;             fraction bit not represented".  What that means is that,
;             since the first bit is always going to be 1, it isn't
;             included.  That's irrelevant for our purposes.
;
; Thus our job here is to extract the sign, exponent, and mantissa,
; and scrunch into 32 bits.
;
sign     = real16[3,*] AND '80000000'XL
exponent = real16[3,*] AND '7FFF0000'XL
mantissa = real16[3,*] AND '0000FFFF'XL


;
; IEEE S_float (REAL*4) format is a 32-bit representation of a float:
;
;   31           23 22           16 15           8 7            0
;    +-+--------------+---------------+---------------+-------- --------+
;   |S|  exponent    |                 mantissa                  |
;    +-+--------------+------------------------------------------ --------+
;
; Note that we have 8 bits of exponent, instead of 15.  Thus, instead
; of excess-16384 (2^14), we have to use excess-127 (2^7).
;
; Note also that we have 23 bits of mantissa.  Since we only get 16
; by masking off the top word of the REAL16, we'll need to shift that
; left and add some more bits from the next word.
;
; Finally, NOTE CAREFULLY that the exponent doesn't live on a nybble
; boundary!  The actual bits used are 0x7F800000 !
;
; IEEE T_float (double, or REAL*8) is a 64-bit representation:
;
;   31                20 19      16 15           8 7            0
;    +-+-------------------+----------+---------------+-------- --------+
;   |S|     exponent       |             mantissa                  |
;    +-+-------------------+---------------------------------- --------+
;   |                       mantissa                            |
;    +---------------------------------------------------------- --------+
;
; Here we have 11 bits of exponent, 20 of mantissa.

IF to EQ 8 THEN BEGIN
  excess    = 1023
  shift_exp = 20
```

```
  ENDIF ELSE BEGIN
    excess    = 127
    shift_exp = 23
  ENDELSE

  ; Convert exponent from excess-16384 to excess-127 or -1024
  exponent = ishft(exponent, -16)
  tmp = where(exponent NE 0, c)
  IF c NE 0 THEN exponent[tmp] = exponent[tmp] - 16383 + excess
  exponent = ishft(exponent, shift_exp)

  ; Add more bits to our mantissa
  IF to EQ 8 THEN BEGIN
    mantissa = ishft(mantissa, 4) + (ishft(real16[2,*], -28) AND '0F'XL)
  ENDIF ELSE BEGIN
    mantissa = ishft(mantissa, 7) + (ishft(real16[2,*], -25) AND '7F'XL)
  ENDELSE

  new_ul = sign + exponent + mantissa
  IF to EQ 8 THEN BEGIN
    new_ul = ishft(ULong64(temporary(new_ul)), 32)
    new_ul = new_ul + (ishft(real16[2,*],   4) AND 'FFFFFFF0'XL)
    new_ul = new_ul + (ishft(real16[1,*], -28) AND '0000000F'XL)

    RETURN, double(temporary(new_ul), 0, N_Elements(sign))
  ENDIF ELSE BEGIN
    RETURN, float(temporary(new_ul), 0, N_Elements(sign))
  ENDELSE
END

 ----snip---------------------------------------------------- ----------
```

--
Eduardo Santiago  Software Type  esm@lanl.gov  RKBA!

---

Subject: Re: 10 bytes real
Posted by Peter Mason on Mon, 30 Oct 2000 21:46:30 GMT
View Forum Message <> Reply to Message

"Karl Schultz" <kschultz@researchsystems.com> wrote:
<...>
> Also, the 80x87 math processors on wintel machines are 80-bit anyway
> and I think that there are instructions that would load 80-bit floats
> into the floating point regs.  After you've done that, you can read
> them back out as a double.  I don't know if there is any C compiler
> support.

<...>

Apparently the MS visual C compiler knows about "long doubles" - 80-bit
IEEE floating point numbers.
*Provided* that your 10-byte numbers are indeed IEEE FP format, you
should be able to use the following routine (compiled etc) to do the
conversion.  On the input side I gues you'd just use a BYTARR or
something, stuffed with the inscrutable 10-byte reals, and for the
output you'd present a DOUBLE array of the right size - hopefully it
will get filled with something useful :-)
I haven't tested the routine below.  (In fact I've never used a "long
double" before.)  But give it a try.
If you don't have a MS visual C compiler, let me know and I will
compile the routine for you.

Cheers
Peter Mason

```
/*
 10-byte to 8-byte IEEE floating-point converter (untested).
 Peter Mason, CSIRO DEM, October 2000
*/
#define STRICT
#define VC_EXTRALEAN
#include <windows.h>

 /************************************************************ ************
******/
BOOL WINAPI DllMain(HINSTANCE hinst, unsigned long reason, void *resvd)
{
 hinst=hinst;  reason=reason;  resvd=resvd;
 return 1;
}
 /************************************************************ **********/
/*
  This is it.
  The call is:
    status=call_external('[pathfconv.dll','idlfp10to8',in10,out8 ,n)
  . in10[n] is an array of 10-byte IEEE floating-point numbers;
  . out8[n] is an array that will be filled with the conversions;
  . n is the number of numbers (in all its splendour?)
*/
int WINAPI idlfp10to8(int ac, int *a[])
{
 register long double *in10;
 register double  *out8;
 register int  n;
```

```
 if(ac!=3) return 1; //incorrect number of arguments
 in10 = (long double *)a[0];
 out8 = (double *)a[1];
 n = *a[2];
 for( ; n; --n, ++in10, ++out8) *out8 = (double)(*in10);
 return 0;
}
 /********************************************************** **********/
```

Sent via Deja.com http://www.deja.com/
Before you buy.

---