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!