Subject: Re: How Computers Represent Floats
Posted by Pavel A. Romashkin on Thu, 30 Nov 2000 08:00:00 GMT
View Forum Message <> Reply to Message

I felt this explanation by Karl was too valuable, so I saved it locally.
I am not sure if this is what you look for, David.
Pavel

Subject:
Re: 10 bytes real
Date:
Fri, 27 Oct 2000 09:39:25 -0600
From:
"Karl Schultz" <kschultz@researchsystems.com>
Organization:
Research Systems Incorporated
Newsgroups:
comp.lang.idl-pvwave
References:
1 , 2 , 3 , 4 , 5

"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.
>
