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

"William B. Clodius" wrote:
> <snip> IEEE 754 requires that all intermediate calculations
> be performed a higher precision so
Ignore the above incomplete sentence. What I originally attempted to write was covered later.
>
> <snip>

Some other surprises.

The definition of the IEEE 754 mantisa, an integer with values from 2^n_mant to 2*2^n_mant-1, where n_mant is the number of bits available for the mantisa, is termed a normalized number. This is error prone for very small numbers. IEEE 754 mandates that there be available for such small numbers what are termed denorms where the mantissa is interpreted as an integer from 0 to 2^n_mant, so that accuracy degrades gradually for such values. However, this complicates the implementation of the floating point, so some processors, e.g., the DEC Alpha make this available only in software at a greatly reduced performance.

The special values for IEEE 754 were chosen with specific applications in mind and are not always the best for other applications. In particular some applications benefit from unsigned zeros and infinities others from signed NaNs none of which are provided by IEEE 754. Only sophisticated users tend to complain about this.

The default rounding behavior for IEEE 754 from an "extended" precision intermediate that is exactly halfway between values in the result representation, always to the same final bit (effectively alternately up and down), often surpises very observant users, although it is designed to reduce the propagation of systematic errors in most applications.

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

Thanks, guys, you are all so very helpful. I have what I need now. :-)

Cheers,

David

P.S. Let's just say I'm going to write an article
for my web page and get this darn piece of information
in a place where I can find it again!

--
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: How Computers Represent Floats
Posted by Phillip David on Thu, 30 Nov 2000 08:00:00 GMT
View Forum Message <> Reply to Message

The result of this search is this link (if it works!):
http://x75.deja.com/ [ST_rn=ps]/getdoc.xp?AN=686544206&CONTEXT=975606363.796262469&hitnum=0

Ivan Zimine wrote:
>
> Studenten wrote:
>>
>> That would be realy nice...Cause i am facing these particular
>> problems...
>> Hope you find the posting...
>> cu
>> Jan
>>
>
> http://www.deja.com/home_ps.shtml
>
> Subject: Re: 10 bytes real
> Date: 10/27/2000
> Author: Karl Schultz <kschultz@researchsystems.com>
>
> --
> Ivan Zimine | ivan.zimine@physics.unige.ch
> Dpt. of Radiology | (+41 22) 372 70 70
> Geneva University Hospitals |

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

Almost every computer newsgroup on programming languages or numerics has this discussion at one time or another. The vast majority of programming languages rely on the hardware representation of floating point numbers so that this issue is essentially programming language independent. The vast majority of computer systems (where computer systems excludes hand held calculators which often implement binary coded decimal) now implement the core of the IEEE 754 standard, so my remarks will be confined to such systems. However the minority of computer systems that do not implement the IEEE 754 standard share many of the same quirks.

Details on these quirks, with an emphasis on the IEEE 754 standard, are available from reports by David Goldberg, "What Every Computer Scientist Should Know about Floating Point Arithmetic," and the subsequent supplement to that report by Doug Priest, "Appendix D". Both reports were written for Sun Computer systems and are available from docs.sun.com in pdf and postscript formats. Links to the documents are available at www.validgh.com.

In almost any binary system including IEEE 754, most of the time floating point numbers can be thought of as divided into three parts, a sign, a mantisa, and an exponent stored in a fixed size "word". In IEEE 754 the mantisa can be thought of as an integer with values from 2^n_mant to 2*2^n_mant-1, where n_mant is the number of bits available for the mantisa. Note that the mantisa is non-zero. In IEEE 754 the exponent can be thought of as a scale factor multiplying the integer, where the scale factor is a simple poser of two whose relative range is determined by the number of bits available for the exponent. IEEE 754 requires that the computer make available to the user two such representations: what we normally think of as 32 bit single and 64 bit double precision. IEEE 754 requires that all intermediate calculations be performed a higher precision so

The fact that the data is stored in a fixed size word results in the first surprise to very inexperienced users: this representation is correct for only a finite number of values. This representation cannot deal with all elements of the countable set of all rationals, let alone the uncountable set of all irrationals. Inaccuracies and errors are almost unavoidable in any attempt to use this data type, except for knowledgeable users in limited domains.

This binary representation does not let IEEE 754 exactly represent numbers that are not simple exact multiples of powers of two. This results in the second surprise to many users: most simple decimal floating point numbers (e.g., 0.3 or 0.1) cannot be exactly represented and any attempt to store such numbers results in errors. E.g., 0.1 might become 0.1000000005 when stored.

Most manipulations of IEEE 754 numbers result in intermediate values

that cannot be represented exactly by single and double precision numbers. This results in the third surprise, most manipulation result in a cumulative loss of accuracy. To minimize this loss of accuracy it requires that intermediate calculations be performed at a higher precision with well defined rounding rules to obtain the final representation of the intermediate results. Most systems appear to use double precision to represent intermediate results for single precision calculations. All systems must use a precision higher than double for intermediate results of double precision calculations. This higher precision representation need not be available to users, however the Intel extended precision is essentially this higher precision intermediate type (it differs slightly in ways that irritate numericists). When this higher precision type is available it need not have the well defined rounding and error propagation propers of single and double precision.

While most of the time IEEE 754 has this behavior there are exceptions represented by special values. Obviously there has to be a zero value. IEEE 754 also has special values such as + or - infinity to represent such things as dividing a finite number by zero, NaN (Not a Number) for representing such things as zero divided by zero, and a signed zero. This allows calcuations to procede at high speed code and the post-facto recognition and (if neecessary) correction of problems with the algorithm or its "inaccurate" implementation in IEEE 754. It also generates floating point exceptions that can be detected automatically withot examining all the output numbers. Many languages were developed before IEEE 754 and do not map naturally to this model.

The existence of infinities and NaN leads to a fourth surprise to many users: obviously bad results can be generated and the computer does not stop the instance it detects such "bad" values. As there are prblem domains where such values are expected and not an indication of problems, it is up to the user to check for such values if they can be generated and when generated indicate a problem.

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.
>

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

Studenten wrote:
>
> That would be realy nice...Cause i am facing these particular
> problems...
> Hope you find the posting...
> cu
> Jan
>

http://www.deja.com/home_ps.shtml

Subject: Re: 10 bytes real
Date: 10/27/2000
Author: Karl Schultz <kschultz@researchsystems.com>

--
Ivan Zimine | ivan.zimine@physics.unige.ch
Dpt. of Radiology | (+41 22) 372 70 70
Geneva University Hospitals |

Subject: Re: How Computers Represent Floats
Posted by John-David T. Smith on Thu, 30 Nov 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Studenten wrote:
>
> That would be realy nice...Cause i am facing these particular

> problems...
> Hope you find the posting...
> cu
> Jan
>
> Nigel Wade schrieb:
>
>> David Fanning wrote:
>>>
>>> Oh, dear. :-(
>>>
>>> I have occasion to recall a discussion posted in this
>>> forum some time ago about how computers represent
>>> floating point numbers. How they appear inaccurate, etc.
>>>
>>> I *know* I saved it, but I've searched on just about
>>> every keyword I can think of on my local machines and
>>> in Dejanews and I can't find what I am looking for.
>>> (It's possible I dreamed the whole exchange. Stranger
>>> things have happened.)
>>>
>>> If anyone saved the discussion (or even recalls it
>>> enough to supply me with some likely keywords to search
>>> on), I would be grateful.
>>>
>>> Cheers,
>>>
>>> David
>>>
>>> P.S. In my dream (apparently) someone who is not a
>>> frequent poster to this newsgroup published a fabulously
>>> informative article on the subject.
>>>
>>> --
>>> 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
>>
>> There was one not too long ago which started with a request
>> about "10 bytes real". Was that the one?
>>
>> If it was back before May '99 then DejaNews won't be able to
>> find it any more.

This brings up the point that it would really be preferrable to have our own indexing, in case Deja News goes under entirely. Anyone have

experience with this?

JD

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

That would be realy nice...Cause i am facing these particular
problems...
Hope you find the posting...
cu
Jan

Nigel Wade schrieb:

> David Fanning wrote:
>>
>> Oh, dear. :-(
>>
>> I have occasion to recall a discussion posted in this
>> forum some time ago about how computers represent
>> floating point numbers. How they appear inaccurate, etc.
>>
>> I *know* I saved it, but I've searched on just about
>> every keyword I can think of on my local machines and
>> in Dejanews and I can't find what I am looking for.
>> (It's possible I dreamed the whole exchange. Stranger
>> things have happened.)
>>
>> If anyone saved the discussion (or even recalls it
>> enough to supply me with some likely keywords to search
>> on), I would be grateful.
>>
>> Cheers,
>>
>> David
>>
>> P.S. In my dream (apparently) someone who is not a
>> frequent poster to this newsgroup published a fabulously
>> informative article on the subject.
>>
>> --
>> 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
>
> There was one not too long ago which started with a request
> about "10 bytes real". Was that the one?
>
> If it was back before May '99 then DejaNews won't be able to
> find it any more.
>
> --
> --
> Nigel Wade, System Administrator, Space Plasma Physics Group,
> University of Leicester, Leicester, LE1 7RH, UK
> E-mail : nmw@ion.le.ac.uk
> Phone : +44 (0)116 2523568, Fax : +44 (0)116 2523555

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

David Fanning wrote:
>
> Oh, dear. :-(
>
> I have occasion to recall a discussion posted in this
> forum some time ago about how computers represent
> floating point numbers. How they appear inaccurate, etc.
>
> I *know* I saved it, but I've searched on just about
> every keyword I can think of on my local machines and
> in Dejanews and I can't find what I am looking for.
> (It's possible I dreamed the whole exchange. Stranger
> things have happened.)
>
> If anyone saved the discussion (or even recalls it
> enough to supply me with some likely keywords to search
> on), I would be grateful.
>
> Cheers,
>
> David
>
> P.S. In my dream (apparently) someone who is not a
> frequent poster to this newsgroup published a fabulously
> informative article on the subject.
>

> --
> 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

There was one not too long ago which started with a request
about "10 bytes real". Was that the one?

If it was back before May '99 then DejaNews won't be able to
find it any more.

--

Nigel Wade, System Administrator, Space Plasma Physics Group,
University of Leicester, Leicester, LE1 7RH, UK
E-mail : nmw@ion.le.ac.uk
Phone : +44 (0)116 2523568, Fax : +44 (0)116 2523555