## Subject: IDL platform difference
Posted by Hubert Dietl on Thu, 25 Mar 1999 08:00:00 GMT

View Forum Message <> Reply to Message

I have noticed that IDL for Windows and IDL for MacOS seem to handle some calculations and/or output using float variables differently. I have a procedure that is carrying out a large calculation and printing the results to a file. When I run the same procedure on the two different platforms, I get two different amswers. Example:

On the PC: 4.94e+009 On the Mac: 4.93e+09

I expected the difference in the number of zeroes in the exponent, but not the change in value. Has anyone ever encountered this before? I understand the difference is small, but there really shouldn't be any difference at all. Thanks for any help.

S.Thiel beorabor@bemail.com

## Subject: Re: IDL platform difference
Posted by Christophe Marque on Tue, 30 Mar 1999 08:00:00 GMT

View Forum Message <> Reply to Message

Peter Mason wrote:

>
> I think that the differences seen here are due to differences in FPU
> architecture (even though all these platforms store the numbers in IEEE
> format *in memory*). The Intel x86 has 80-bit floating-point registers. I
> don't know what mac hardware you're using, but I'd bet that it's different.
> From what I recall, the Alpha is less than 80 bits. (I think you get what
> you ask for on Alpha? - viz. 32-bits for single precision.) So results for
> an operation as simple as a single subtraction or addition can differ
> noticeably, even in double precision. Even a half-decent compiler will keep
> some operands in registers for a while (at least sometimes), so these
> differences can easily build up. (Well, there's a compiler option to force
> them out to memory straight away, but I don't think IDL is compiled like
> this.) So given the nature of the beast, it's risky to rely on *exact*
> floating-point numbers, especially across platforms and/or with algorithms
> that push the precision.
>
> Peter Mason

The problem seems to be more general: we upgraded from IDL5.0 to IDL5.2 (for the UNIx Platform). Numerical results are different for IDL 5.2 and IDL5.0 on the same platform. These new results are closer to WINDOWS results(5.0).

When we use IDL 5.2(windows) with double floats we obtain the same
results on all IDL5.2 platforms.
Does someone else encounter the same troubles in upgrading from 5.0 to
5.2?
Thank you
--
Christophe Marque

## Subject: Re: IDL platform difference
## Posted by menakkis on Tue, 30 Mar 1999 08:00:00 GMT

View Forum Message <> Reply to Message

> Hubert Dietl wrote:
>> I have noticed that IDL for Windows and IDL for MacOS seem to handle
>> some calculations and/or output using float variables differently. I
>> have a procedure that is carrying out a large calculation and printing
>> the results to a file. When I run the same procedure on the two
>> different platforms, I get two different amswers.
<...>

And Christophe Marque <Christophe.Marque@obspm.fr> wrote:
> I have yet encountered the same problem in running a complicated program
> on a Windows NT IDL and a UNIX Digital alpha IDL.
<...>
> I thought the difference was the 2 kinds of processors:
> Intel 32 bits and Dec alpha 64 bits.
> The main trouble is you have the same behaviour when you use double
> floats instead of floats.

I think that the differences seen here are due to differences in FPU
architecture (even though all these platforms store the numbers in IEEE
format *in memory*). The Intel x86 has 80-bit floating-point registers. I
don't know what mac hardware you're using, but I'd bet that it's different.
From what I recall, the Alpha is less than 80 bits. (I think you get what
you ask for on Alpha? - viz. 32-bits for single precision.) So results for
an operation as simple as a single subtraction or addition can differ
noticeably, even in double precision. Even a half-decent compiler will keep
some operands in registers for a while (at least sometimes), so these
differences can easily build up. (Well, there's a compiler option to force
them out to memory straight away, but I don't think IDL is compiled like
this.) So given the nature of the beast, it's risky to rely on *exact*
floating-point numbers, especially across platforms and/or with algorithms
that push the precision.

Peter Mason