The x86 architecture is the weirdo: Structured exception handling

devblogs.microsoft.com/oldnewthing/20130320-00

March 20, 2013

Raymond Chen

If your reference architecture is x86, then you will think that everything it does is normal and the rest of the world is weird. Except it's the other way around: The x86 architecture is the weirdo.

I was reminded of this when commenter 640k complained, on the subject of what I think is table-based structured exception handling, "It would be interesting to know why this 'invention' was introduced in 64-bit Windows when no other version of Windows requires it." (The original text was "when no other OS requires it", but I'm assuming that this was in the context of Windows-based OS, since unix doesn't have structured exception handling in the first place.)

This has a very narrow definition of "no other OS", because it really means "No other non-x86-based version of Windows." In this world, the color of the sky is x86.

In fact, x86 is the *only* architecture for which Windows uses stack-based exception chaining. All other architectures use table-based exception unwinding. The prologue and epilogue of each function must follow a particular format so that the actions performed therein can be unwound during exception handling. At the very introduction of Win32, it was only the x86 which used stack-based unwinding. The Alpha AXP, MIPS, and PowerPC all used used table-based exception unwinding. And as new architectures were added by Windows, they all used table-based exception unwinding as well. Itanium? Table-based. Alpha AXP 64-bit? Table-based. ARM? Table-based.

The use of table-based exception handling was not "introduced" with x64. It was introduced back in 1992, and has in fact been the exception unwinding mechanism for all architectures.

Well, almost all. Not the x86, because the x86 is the weirdo.

Raymond Chen

Follow