Why did InterlockedIncrement/Decrement only return the sign of the result?

devblogs.microsoft.com/oldnewthing/20040506-00

May 6, 2004

Raymond Chen

If you read the fine print of the InterlockedIncrement and InterlockedDecrement functions, you'll see that on Windows NT 3.51 and earlier and on Windows 95, the return value only matches the sign of the result of the increment or decrement. Why is that?

The 80386 instruction set supports interlocked increment and decrement, but the result of the increment/decrement operation is not returned. Only the flags are updated by the operation. As a result, the only information you get back from the CPU about the result of the operation is whether it was zero, positive, or negative. (Okay, you also get some obscure information like whether there were an even or odd number of 1 bits in the bottom 8 bits of result, but that's hardly useful nowadays.)

Since those operating systems supported the 80386 processor, their implementations of the InterlockedIncrement and InterlockedDecrement functions were limited by the capabilities of the processor.

The 80486 introduced the XADD instruction which returns the original value of the operand. With this additional information, it now becomes possible to return the result of the operation exactly.

Windows NT 4 dropped support for the 80386 processor, requiring a minimum of an 80486, so it could take advantage of this instruction. Windows 98 still had to support the 80386, so it couldn't.

So how did Windows 98 manage to implement an operation that was not supported by the CPU?

Windows 98 detected whether you had a CPU that supported the new XADD instruction. If not, then it used an alternate mechanism which was mind-bogglingly slow: It called a driver whenever you wanted to increment or decrement a variable. The driver would then emulate the XADD instruction by disabling interrupts and performing the operation in locked memory. Since Windows 98 was a uniprocessor operating system, it didn't have to worry about a second processor changing the memory at the same time; all it needed to ensure was that the single processor didn't get interrupted while it was performing the "atomic" operation.

Raymond Chen

Follow