

Cache Coherency

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.development.device.drivers/2006-06/msg01356.html>

- *From:* mark.moore@xxxxxxxxxxxxxxxx
 - *Date:* 19 Jun 2006 10:07:32 -0700
-

In an earlier post[1], Stephan Wolf says "On x86, there is no need to take care of any caching issues at all because x86 always guarantees cache coherency."

I may be taking him too literally, but I don't see how that can be true. I've got a PCI adapter using a PLX 9030 that presents a 1K memory buffer at PCI BAR4 which is marked as cacheable to the OS. This is mapped in at DriverLoad using MmMapIoSpace() with MmCached.

Once the OS is done transferring data out of the buffer, the buffer is released back to the adapter to refill with incoming data. When the memory is done refilling, the adapter hands ownership of the memory back to the OS and an interrupt is asserted.

The problem I'm trying to address is how to ensure the CPU reloads its internal cache with the new data since I can see no way that it can be aware of the changes. The PLX 9030 is a target only device, so it can't master a special cycle on to the system bus. And I didn't see any way the CPU could snoop the memory for changes.

The natural (and documented) way to keep the cache coherent is to call KeFlushIoBuffers() in the IRQ DPC before reading the buffer. But, as Stephan mentions, and as I've confirmed, this is completely macrolized away for x86 builds. It doesn't do anything.

I haven't seen any HAL'd abstractions in the MmXxx() or KeXxx() routines that invalidate specific cache lines, so I think I'm being forced to get a little closer to the metal.

It looks like the right thing to do would be to issue an CLFLUSH[2] op sandwiched between two MFENCE ops. But, that's pretty brute force, and I'm not sure these are supported by AMD. At least I couldn't find a definitive reference. If they definitely are, or if there's a different way to flush a cache line on AMD, let me know.

Of course, WBINVD will work, but that flushes the whole cache to system memory, and all I need to make coherent is a tiny chunk that's <1K. Mighty inefficient.

Cache Coherency

Another alternative would be to issue an INVLPG, but that seems to get all over the Memory Manager's business. If this were the right way, it would probably be better to MmUnmapIoSpace() and then re-map them.
<gack>

Hopefully this wasn't too newb. I've looked around the net and here for answer, so fire away!

-Mark

[1]

http://groups.google.com/group/microsoft.public.development.device.drivers/browse_frm/thread/7b6104ea25dfcebb/3

[2] Sure wish there was a MmFlushCache(base, size, read_not_write) call. Wait! There's KeFlushIoBuffers() now. :-/

.