

## Re: x86 architecture questions

**Source:** <http://coding.derkeiler.com/Archive/Assembler/comp.lang.asm.x86/2004-03/0542.html>

---

**From:** Matt Taylor ([para\\_at\\_tampabay.rr.com](mailto:para_at_tampabay.rr.com))

**Date:** 03/26/04

Date: Fri, 26 Mar 2004 16:19:42 +0000 (UTC)

"Scott Moore" <[samiam@moorecad.com](mailto:samiam@moorecad.com)> wrote in message  
news:UqQ8c.95875\$Cb.1259693@attbi\_s51...

<snip>

> See other replies in this thread. You don't need the privileges in  
> paged management based code. The best way is if the kernel does not  
> even reveal any of its code to the target (user) processes. Each  
> run in their own virtual space. There is no "key" to protected space  
> required, because such space is not mapped in common to user space  
> unless the kernel wants that.

This ignores the fact that OSes also run a 3rd type of code: drivers. The 4-ringed system was designed to incorporate drivers. Drivers need a moderate level of protection; it makes sense for them to access OS data structures, but you typically don't want them to modify them. The only alternative is to run drivers in ring-3.

I'm pretty sure QNX actually runs drivers in ring-3, and I know Microsoft is planning to. If Intel had used 4 rings in the page table entries, life would be much simpler. The tradeoff for drivers is performance (ring-0) or stability (ring-3). Using ring-1 for drivers in this case would make systems much more stable without compromising performance.

It would also make a lot of sense if Intel had implemented R/W and U/S perms as 2 fields of 2 bits determining the lowest privileged ring which could read or write respectively. It is possible to map the same memory into multiple places in the address space in order to allow the kernel R/W access and the user only read access, but it is a waste of space. I don't know what the tradeoff is in silicon, but I don't think it would be much worse than what was done.

> Yes, I am serious. Those features are clearly on their way out.  
> Intel is moving them to the emulation unit, which means they are  
> slower, perhaps a lot slower, and Intel is deprecating them. Keep  
> in mind, Intel is even now recommending that you set up your OWN  
> task state tables and switching code. That's a big change for them.

Indeed. Intel has been all too eager to move OS functionality into silicon,

not vice versa.

- > *Whats really odd is (to my understanding) Windows uses do-it-yourself*
- > *TSS and Linux uses the hardware supported TSS. You would think it*
- > *would be the opposite.*

I would expect both NT and Linux to do it themselves as they are both intended to port to RISC architectures. I would expect Win9x to use TSS. I have been told Win9x uses call gates.

- > *Bottom line: IBM pushed virtual memory in the 360 product line, and*
- > *MIT thought up segments and rings as a cheaper solution. Virtual*
- > *memory won. Its over.*

The theoretical term "ring" may be tied to segmentation, but on Intel a ring is the current privilege level. It is used by both the paging and segmentation systems. It is inseparable from the architecture itself.

-Matt