THE EDITORS' VIEW

What's Next for Intel?

A "Trojan Horse" Strategy for Converting to RISC

As I described in my March 29 editorial, Intel's Pentium processor reinforces the argument that RISC architectures have a significant technical advantage. Whether you look at die size, transistor count, design time, or peak performance, Pentium fares poorly when compared to most of its RISC competitors. That Pentium performs as well as it does is a testament to the ability of Intel's engineers, but just think what they could do if they were not hobbled by the limited and baroque x86 architecture. With the emergence of Windows NT and the prospect that application developers might begin supporting multiple architectures as a matter of course, the architectural weaknesses of the x86 could turn into a serious disadvantage in the marketplace.

As much as Intel talks about "x86 forever"—and you could hardly expect them to say anything else—the Pentium vs. RISC comparison surely must not be lost on Intel's planners. Indeed, there are widespread rumors that major changes will be made to the programming model in P6 and P7, with a larger register file and possibly an entire new instruction set.

Such a transition is likely to come in concert with the evolution to a 64-bit architecture. At last year's Microprocessor Forum, Intel's Pat Gelsinger commented that, just as Intel enhanced the architecture along with making the 16- to 32-bit transition in the 386, a future 64-bit version would also introduce another set of architectural improvements, including a more regular instruction set. While Intel might just add 64-bit features and other extensions on top of the existing 386 instruction set, the appeal of getting rid of the variable-length instructions and limited 32-bit register set is clear.

If we assume that Intel has the desire to move to a more modern instruction set, the question becomes, "How can they do it?" Clearly even Intel cannot simply introduce a new architecture and declare it the successor to the x86, for this would open up the market to all the RISC competitors. Motorola did this with the 88000, and as a result they prodded the workstation market to abandon the entrenched 68000 family for a variety of other RISC processors.

The emergence of other 386 and 486 processor vendors provides additional incentive for Intel to change the architecture, but it might also make it more difficult. Just as IBM no longer drives the PC standard it created, Intel might not be able to drive the x86 standard in the future—though for now, Intel probably still has enough clout to do so.

Intel's opportunity, it seems, is to create a gradual transition by introducing a new architecture while retaining compatibility with the x86. One way this might work would be for a next-generation x86 chip—the P6 perhaps, or if not the P7—to be based on a pure, RISC-style core, with on-chip interpretation hardware to translate x86 instructions into core instructions. (Pentium, and even the 486, are sometimes described in this way, but it is not accurate.) This may not be as fast or efficient for x86 code as a pure x86 machine like Pentium, but it would have one major advantage—the native instruction set of the RISC core could be used by new software. Such a chip would, in effect, support two instruction sets: the traditional x86 instruction set for compatibility, and a new, RISC-like instruction set for maximum performance.

Intel could establish a new architecture with such a chip using a "Trojan horse" strategy. Initially, customers would buy this chip simply as the next-generation x86 processor, and little if any software would support the native mode. The first native-mode support might come in operating-system code; Windows NT, for example, might use the chip in native RISC mode for operating system functions and switch to x86 mode for applications. As long as the chip was cost-effective as an x86 processor, Intel could sell millions of the hybrid chip without needing widespread support for the new instruction set. Intel might accept lower-than-usual margins to drive acceptance of the hybrid device.

Once this installed base was created, application developers would be willing to support the new, native instruction set to get the performance gains. Then, for the following generations of chips, Intel could drop the on-chip hardware support for x86 emulation, substituting a software emulator to support older programs that didn't support the new architecture.

At this point, Intel would have made the transition to a new architecture in a relatively non-disruptive manner. Furthermore, Intel would have left behind all the x86-compatible chip makers, which would then be stuck with an obsolete instruction set. While they could eventually follow, Intel would surely find ways to make it hard for them, thereby extending its control over the high-end PC microprocessor market. ♦