Cahners
MICROPROCESSOR REPORT
www.MPRonline.com
THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

AMD Drops 64-Bit Hammer on x86

Could 64-bit Extensions Stretch the x86 Presence Another Decade?

By Steve Leibson {9/4/00-01}

With IA-64 (Itanium) already working its way through the prototype cycle, it seems almost certain that Intel has finished with the 20-year game of x86 enhancement. But AMD has not. At last year's Microprocessor Forum, AMD's Fred Weber announced that the company intends to create a 64-bit extension to the x86 architecture (code-name Hammer). Now, the company has made a programmer's overview of the extended x86 64-bit architecture, formally dubbed x86-64, available on AMD's Web site at www.amd.com/products/cpg/64bit/overview.html. The x86-64 programmer's overview is not a full processor specification, but it's certainly enough to get a sense of how AMD plans to extend the x86 architecture into a 64-bit future. AMD's plan is to extend the x86 architecture into the 64-bit realm without breaking the 32-bit model, in much the same way that the '386 transported the 8086 from the 16-bit world to the 32-bit world in 1985. The company expects to introduce the first processors based on x86-64 at the end of 2001. In addition to the programmer's overview, AMD is developing an x86-64 compiler. The compiler and an instruction-set simulator should be available later this year.

Easier Than an Itanium Break

The reasons for AMD's 64-bit x86 strategy are obvious. More software runs on the x86 architecture than on any other processor on the planet. Although most of this software is relatively uncramped in the 32-bit address space of the x86 architecture, as currently evolved, the existing 4GB limit is starting to pinch data-heavy applications like large databases, streaming video, mechanical CAD and electronic design automation (EDA), security and encryption, and Internet infrastructure. These applications must wrestle with huge data structures that can easily exceed 4GB. Systems developers working in these application areas are eagerly evaluating the many 64-bit architectures and processors already available from ARM, IBM, MIPS, and Intel's IA-64. The realm of 64-bit computing is currently terra incognita for x86. Hence, AMD senses an opportunity.

AMD says the x86-64 extensions will allow it to pursue its goal of producing the fastest x86 processors available, and it expects the x86-64 extensions to add only 5% or less to the die size of its next-generation x86 processor. The company therefore claims that these 64-bit extensions will not inflict a cost or speed penalty on its next x86 generation. AMD's 64-bit architectural strategy consists of six essential extensions to the existing x86 architecture:

- Full backward binary compatibility with existing 32-bit x86 instructions
- A 64-bit instruction-set extension
- A 64-bit register-set extension
- An enhanced memory model that includes 64-bit addressing extensions
- A mode bit in the extended feature enable register to enable 64-bit operation
- An additional bit in the code-segment descriptor table to tag 64-bit code

Too Many Extensions for Full Compatibility?

Perhaps the first element, full backward compatibility with the existing 32-bit x86 architecture, is the most ambitious, because there are so many extensions to the original x86 32-bit

architecture as defined by the '386, and AMD has never implemented some of them. Although the x87 floating-point and MMX multimedia extensions are now staples of all x86 derivatives, AMD's processor designers must also add AMD's own 3DNow! (admittedly, not much of a stretch) and Intel's SSE and SSE2 to be truly compatible with all the x86 processors that preceded the x86-64 generation.

As Figure 1 shows, the x86-64 extension to the general-purpose register set is straightforward and badly needed. Just as the '386 widened the 8086's original 8 registers from 16 bits to 32, x86-64 widens them yet again, to 64 bits. However, the x86-64 register extension also provides 8 additional 64-bit registers, called R8 through R15, for a total of 16 registers, although not all of them can be considered general purpose. Although 16 registers might be a goodly number for an x86 CISC processor riding around on a 20-year-old instruction-set chassis, it's still a small number compared with those in Intel's IA-64 and many RISC architectures. By comparison, the IA-64 specification calls for 128 general-purpose registers and 128 floating-point registers, in addition to several other miscellaneous registers. The MIPS, PowerPC, and Alpha architectures all have 32 general-purpose registers, and ARM10 has 31, with 15 visible at any given time. Modern compilers

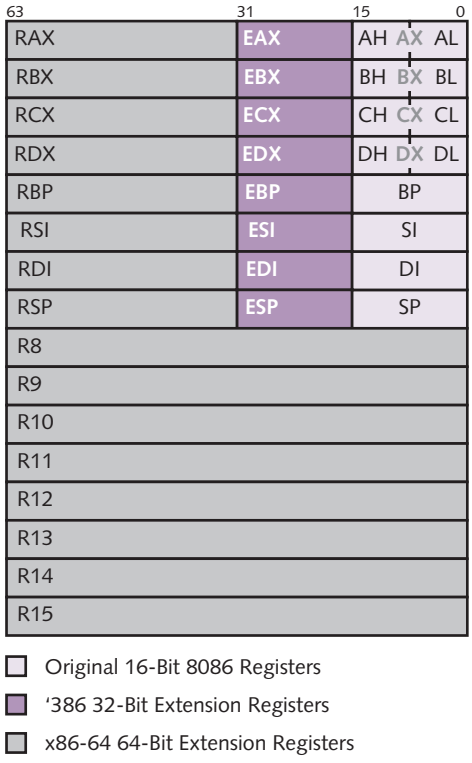

Figure 1. AMD's x86-64 specification widens existing x86 general-purpose registers to 64 bits and adds another eight registers to the bank. The x86-64 register set also includes the eight x87/MMX and eight 128-bit SIMD SSE registers defined by previous extensions to the original x86 register set, as well as an additional eight SSE registers (for a total of 16).

can really take advantage of larger register files to speed code execution by reducing the amount of traffic between registers and memory. To be fair, AMD says that code traces on existing software indicate that the additional 8 general-purpose registers produce the lion's share of performance benefit, and that adding registers beyond the x86-64's 16 would provide diminishing returns at the cost of significant additional complexity. Of course, none of this matters a whit to compilers dealing with 32-bit x86 code, because the only registers those compilers can ever use are the original 8 on the x86. In addition to 8 more 64-bit registers for the general-purpose set, x86-64 also appends 8 more 128-bit registers to the SIMD SSE register set, for a total of 16.

The x86-64's 64-bit instruction-set extension mostly involves extending the existing x86 instructions to handle 64-bit data types. Instructions operating on 64-bit operands contain a REX prefix inserted into the x86 instruction stream, as shown in Figure 2. All 32-bit x86 applications run unchanged, using the existing x86 instructions, while software compiled using the x86-64 compiler's 64-bit mode can take advantage of the new instructions that handle 64-bit data types. Also added to the 64-bit instruction set is a new set of IEEE floating-point instructions that supplement the existing x87 floating-point instruction set. These new x86-64 floating-point instructions use the existing SSE/SSE2 registers. The SSE registers are configured as a flat register file, in contrast to the stack-based x87 floating-point registers, so x86-64 floating-point code should prove considerably faster than similar code written for 32-bit x86 processors.

Long-Mode Access Is the Key

Augmenting the 32-bit x86 memory model is somewhat more difficult. Naturally, existing 32-bit programs will not be able to understand or use the full 64-bit address space. The x86-64 specification adds a bit, called the "L" bit (for long mode), to the code segment descriptor (CSD) table, so that the 32/64-bit operating mode is selected on a code-segment (application) basis. Another new bit, added to the extended feature enable register (EFER) and dubbed the LME (long-mode enable) bit, enables 64-bit operation when it is set. However, the processor doesn't enter into long mode until software enables paging subsequent to asserting LME. When the LME bit is set and paging is then enabled, the x86-64 processor asserts the read-only LMA (long-mode active) bit in the EFER, signaling that the x86-64 processor has entered long mode.

The LMA bit, along with the new L and existing D bits in the CSD table, determines the memory-addressing mode used for each code segment, as shown in Table 1. When LME is zero, the x86-64 processor uses 32-bit addressing and ignores the L bit in the CSD. During 32-bit operation, the processor operates in x86 real mode if the code segment's D bit in its CSD table entry is "0." If the D bit is a "1" while LMA is "0," the processor operates in the x86's 32-bit protected mode.

Setting LMA to 1 enables long-mode 64-bit operation. However, 32-bit code can still execute when the processor is

in 64-bit mode. If the code segment's L bit is 0 (denoting a 32-bit x86 code segment), the x86-64 processor running in long mode executes the 32-bit code segment in compatibility mode. In this mode, 32-bit code has access to a separate 32-bit address space carved from the x86-64's larger 64-bit address space. If the L bit and the LMA bit are both set to 1, the x86-64 processor operates in 64-bit mode. It is only in this mode that the application code can see the full set of 64-bit register extensions and use the new 64-bit instructions. Note that AMD has set aside and reserved the case where the LMA and the CSD L and D bits are all set to 1.

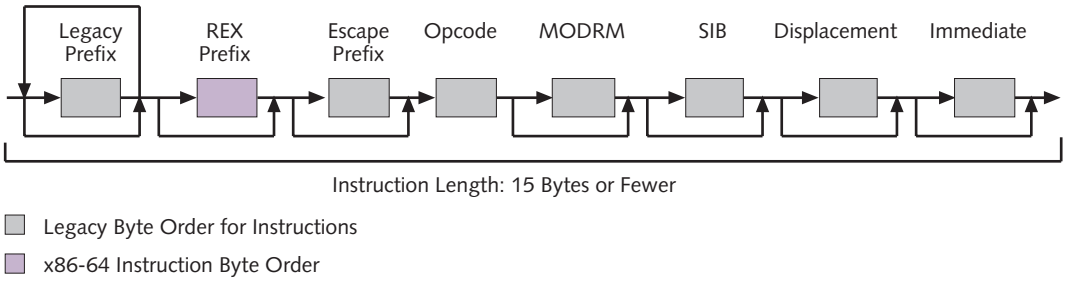

Figure 2. The x86-64 specification adds another byte, called the REX prefix, to the x86 instruction byte sequence to denote instructions using 64-bit operands.

In the case where the x86-64 processor runs a 32-bit operating system, LMA is never set, and the processor operates like any other 32-bit x86 processor. However, if the x86-64 processor is running a 64-bit operating system, it will handle all interrupts and exceptions in long mode while running applications in the appropriate mode.

Long Live the x86

Enhancing the x86 architecture has been a 20-year experiment in processor life-extension. Every time the old x86 locomotive appears to run out of steam, an architectural enhancement seems to bump up the pressure in the boiler. It almost didn't happen. The original 8086 had a 16-bit architecture. The 80186 was not fully compatible with the only software that mattered for that processor (Microsoft's DOS), and the '286 memory-management enhancements were so bad that no one speaks of them. The immensely successful '386 enhancements propelled the x86 into the realm of 32-bit viability by adding paged memory management and widening the general-purpose registers to 32 bits.

Since then, processor designers have focused on executing the '386 instructions as fast as possible by adding cache (the '486) and then multiple execution units (Pentium). Finally, x86 processor designers have broken with the past by creating processors that shred x86 instructions into micro ops and then schedule several of these lower-level instructions simultaneously for execution in superscalar pipelines (Pentium Pro, II, III, and 4, and Athlon) that bear no resemblance to the original x86 microarchitecture. True, there have been some architectural bolt-ons, such as the x87 FPU (starting with the '486), MMX and MMX2, and SSE and SSE2. However, these extensions are intended for handling floating-point computations or multimedia datastreams, and they conveniently sidestep the fundamental limitations of the original x86's cramped register set by making entirely new and separate sets of registers available to the new instructions.

Intel has chosen to let the x86 series remain a 32-bit architecture (which Intel calls IA-32) and to develop IA-64 as its entry into the 64-bit realm. IA-64 is an entirely new 64-bit architecture, with a completely different instruction set. IA-64 melds RISC and VLIW concepts to create an architecture that Intel calls EPIC (explicitly parallel instruction-set computing). EPIC relies on the software compiler to find inherent instruction-level parallelism in an application program, and to exploit that discovered parallelism by creating instruction groups that the processor can execute in tandem. This approach moves the complexity of finding instruction-level parallelism out of the silicon and into the compiler. Presumably, the compiler can use its greater scope to find and exploit more parallelism in the entire application program. A processor has a much more limited scope for discovering parallelism through its relatively small reorder buffer. Software optimization and scheduling also allows processor designers to simplify processor pipelines by eliminating the need for pipeline stages that track instruction dependencies—because the compiler has already handled the scheduling long before run time.

For now, Intel has committed to making every IA-64 processor compatible with IA-32 binaries, thus maintaining backward software compatibility. Itanium processors have two instruction decoders: one for IA-64 instructions and one for IA-32 instructions. A processor mode bit selects the decoder that will receive the instruction stream from memory. Special branch instructions set and clear this mode bit, so the actual machine instructions do not carry IA-64/IA-32 encoding information and cannot be identified as IA-64 or IA-32 instructions, except in context. The separation of

Mode	LMA	CSD L Bit	CSD D Bit
Normal 16-Bit	0	X	0
Normal 32-Bit	0	X	1
Compatibility 16-Bit	1	0	0
Compatibility 32-Bit	1	0	1
64-Bit	1	1	0
Reserved	1	1	1

Table 1. The x86-64 spec brings 64-bit operation into the x86 memory model compatibly. In 64-bit mode, an x86-64 processor can still run existing x86 protected- and virtual-mode programs based on each code segment's descriptor-table entry.

Price & Availability

The x86-64 Programmers Overview is available now on AMD's Web site at www.amd.com. Information on the availability of a processor based on this specification has not yet been released.

instruction sets allows Intel to reuse the entire code space for IA-64 instructions without fear of conflicting with IA-32 instructions.

Is Resistance Futile?

With the introduction of the x86-64 specification, AMD clearly signals that it plans to diverge from Intel in the desktop, workstation, and server processor markets. In the 1980s, AMD often grabbed the lead in x86 development by offering faster processors than Intel did, within the same architectural generation. This practice ended with Intel's introduction of the '486 in 1989. AMD's x86 processor performance didn't catch up to Intel's until AMD introduced Athlon last year, making the desktop-processor game a horserace once more. It now seems clear that Intel and AMD will again diverge—and soon. Intel's Pentium 4 series could be the last of the x86 line for Intel. By the time the Pentium 4 core runs out of gas, Intel expects to have IA-64 firmly entrenched. If IA-64 and 64-bit computing do become pervasive, Intel will have far less impetus to again refresh the Pentium line. AMD believes the x86 architecture has plenty of steam left and has set out on a different path—one that allows a more gradual, and possibly more orderly, transition to 64-bit computing. However, without real Hammer processors to evaluate, the performance potential of AMD's x86-64 approach cannot be evaluated.

Unlike the armies of companies innovating in the MIPS and ARM spaces, neither AMD nor Intel has signed on another processor vendor to join the journey to 64 bits, although Intel does have HP as an architectural-development partner. Neither vendor has really worked with other semiconductor partners, or with each other, in the processor arena for more than a decade. Both are accustomed to going it alone architecturally and competing mightily against each other.

On the operating-system front, Intel has lined up Microsoft Windows, Linux, AIX, and HP-UX for IA-64. According to AMD, the legion of Linux has already jumped on the x86-64 spec. One would assume that AMD is also trying very hard to woo Microsoft.

Intel's IA-64 has already captured the allegiance of nearly every workstation vendor except Sun, and thousands of working Itanium processors are already in the hands of OEMs and developers. AMD's x86-64 entry is far newer, and no silicon yet exists. In many ways, however, x86-64 represents a much more conservative approach and should be attractive to high-end PC manufacturers looking to extend their product lines for a few more years. It seems unlikely that AMD's x86-64 will dent Intel's workstation and server alliances. However, the PC vendor community is virtually uncommitted to 64-bit computing for now, so the game is still afoot. ◇