VOLUME 12, NUMBER 13
OCTOBER 5, 1998

MICROPROCESSOR REPORT

THE INSIDERS' GUIDE TO MICROPROCESSOR HARDWARE

Katmai Enhances MMX

Intel Gives Peek at New SIMD Floating-Point Architecture

by Keith Diefendorff

In an unusual reversal of roles, Intel for the first time has been racing to catch up with AMD on x86 architectural extensions. But catch up it has, or soon will. At its recent developer forum (see MPR 10/5/98, p. 16), Intel opened the Katmai kimono—however slightly—to show its floating-point extension to the previously integer-only MMX architecture. The extension, known only as Katmai new instructions (KNI, alias MMX2), will ship on the eponymous processor in 1Q99, more than six months after AMD shipped K6-2 with its similar but less powerful 3DNow (see MPR 6/1/98, p. 18).

The KNI extension is more than just new instructions. In a move best described as "about time," Intel has added new registers to the x86. The registers are the first new architectural state added since 1985, when the 386 extended the architecture from 16 bits to 32. While most of the 70 KNI instructions deal with floating-point SIMD (single instruction, multiple data) operations, KNI also adds a few new integer MMX instructions and, for the first time, adds explicit cache-prefetch instructions—a feature common to RISC architectures. Intel sill has not, however, disclosed full details of the KNI instruction set.

SIMD FP Boosts Multimedia Performance

SIMD-style processing is uniquely well suited to digital signal processing in multimedia applications, due to the high degree of data-level parallelism inherent in the underlying algorithms. Where data parallelism exists, vector architectures are the obvious choice. SIMD is similar but breaks large vectors into sequences of short, fixed-length vectors, making SIMD easier than a full vector pipeline to add to a general-purpose scalar processor.

Today, floating-point (FP) in multimedia processing is mainly limited to 3D-graphics geometry calculations. Other multimedia applications use fixed-point arithmetic almost exclusively. In part, this is because a lot of multimedia data is naturally fixed-point; for example, color pixels are four-element vectors of 8-bit integer intensity values. But many multimedia tasks—audio and speech processing, for example—are actually better suited to floating-point. They are restricted to fixed-point because of the performance limitations of most floating-point hardware, or, rather, the high cost of sufficiently fast floating-point hardware. KNI could change this.

Intel is the fourth processor vendor to announce a SIMD floating-point extension to its general-purpose processor architecture. This trend began with MIPS V (see MPR 11/18/96, p. 24) and continued with AMD's 3DNow, and then with PowerPC's AltiVec (see MPR 5/11/98, p. 1). Together, these four adopters could—finally—represent a large enough commitment to floating-point to motivate software vendors to adopt floating-point algorithms.

New Registers, New Instructions

As Figure 1 shows, KNI introduces eight new registers, XMM_0–XMM_7. Each 128-bit register holds a vector of four IEEE single-precision floating-point elements. Unlike the existing scalar FP register file, the new register file is directly addressed, dispensing with the performance burden imposed by the stack-based address model of the former.

KNI's instruction set includes all the standard arithmetic operators (+, –, ×, ÷, square root, absolute value, and

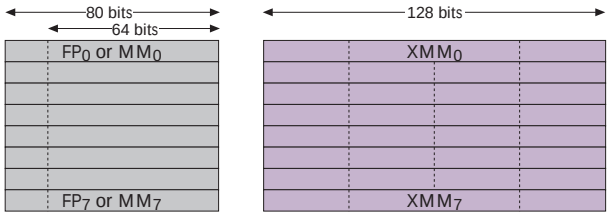

Figure 1. For the first time since the 386, Intel has added to the x86's architectural state with eight new registers. The registers are each 128 bits wide, holding four IEEE single-precision floating-point values. Unlike the scalar FP registers, the new register file uses a simple direct-addressing scheme.

negate). As in 3DNow, KNI also provides reciprocal and reciprocal–square-root estimates, which require Newton-Raphson refinement to achieve IEEE accuracy.

As Figure 2 shows, KNI's SIMD-FP instructions take two source-vector operands from the new register file, perform the specified operation on respective elements, and return a result vector to the new register file. Most of the instructions are also available in scalar form, which operates on only the least significant element of the operand vectors.

Compare instructions are provided, but Intel has not yet described these instructions. Presumably, like the MMX compares (see MPR 3/5/96, p. 1), they generate a boolean result vector that is used by subsequent instructions to control movement of selected elements from one register into another. In MMX, this movement is accomplished by sequences of boolean operations. Although KNI provides no conditional move, some undisclosed capability is provided for this function. Even though this feature eliminates many branches, KNI compare instructions also set condition-codes for the cases in which data-dependent branches cannot be avoided.

New load and store instructions move vectors between the new registers and memory. Intel did not describe these instructions completely but did say that data transfers are always aligned on even 128-bit address boundaries in memory; misaligned data is extracted in the registers by separate instructions. The new loads and stores use the same addressing modes as existing x86 memory-referencing instructions.

Special "swizzle" instructions provide a means for rearranging elements within a vector—an important performance feature in SIMD architectures. KNI apparently provides commonly used transformations like packing, unpacking, transposition, and alignment, but it does not offer the more flexible rearrangement capability provided by AltiVec's permute function.

KNI performs floating-point arithmetic in either a fully IEEE-754–compliant mode or in a flush-to-zero (FTZ) mode. In FTZ mode, underflows return zero rather than an IEEE denormal result. In many applications—such as 3D—the graceful underflow afforded by denormals is not required, and the faster operation of FTZ mode is more valuable. In contrast, 3DNow provides only FTZ operation, making it unsuitable for applications that do require full IEEE arithmetic. Intel did not say where KNI's IEEE exception flags and rounding control bits are stored, but presumably there is a new status and control register for this purpose.

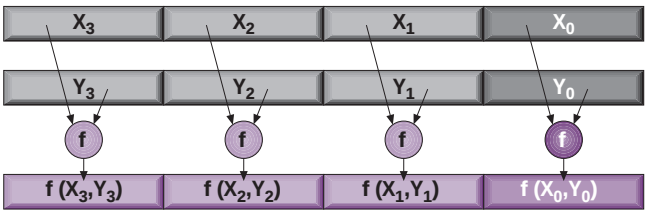

Figure 2. KNI's SIMD-FP instructions take two source vectors (gray), perform the same operation (f) on all four single-precision elements, and return the result vector (purple). Scalar-form instructions operate on only the least significant element (dark).

Floating-point vectors in the new registers can be converted to integers and stored in the MMX registers, and vice versa for integer to floating-point. If KNI uses 32-bit precision for the integers, as we suspect, a pair of MMX registers will be required to hold a full vector of four integers. Register pairing requires two destination-register reservations, complicating dependency analysis in the instruction issue logic, which could potentially impact cycle time.

Although Intel did not disclose implementation details of Katmai, it did say that its peak processing rate will be 2 GFLOPS at 500 MHz. To minimize hardware, Katmai can execute only half of a vector multiply (two multiplies) and half of a vector add (two adds) per cycle. Thus Katmai cannot sustain back-to-back vector multiplies, a significant sacrifice in throughput from that provided by the KNI architecture. This compromise introduces an ugly scheduling constraint and limits Katmai to the same multiply and add throughput per cycle as K6-2 with 3DNow, although Katmai will achieve higher clock rates.

KNI Beefs Up Integer MMX

Several new integer MMX instructions were added to address shortcomings of the original MMX set relative to other vendors' media extensions. One is MMX's lack of assist for motion estimation, a critical requirement for MPEG encoding. To support this function, KNI adds a sum-of-absolute-differences (SAD) instruction that computes $\sum|X_i-Y_i|$ for each of the four 16-bit integers in two 64-bit MMX vectors. SAD is important to the task of searching for similar macroblocks in adjacent video frames for motion-vector coding. KNI's SAD improves search performance by 60% over MMX. Also provided is a complementary packed-average instruction, $(X_i+Y_i+1)/2$, for motion compensation in MPEG decoding.

KNI also adds minimum and maximum instructions that are useful in a number of signal-processing algorithms but were added to KNI primarily to support Viterbi search in speech-recognition algorithms.

KNI Increases Bandwidth

Many multimedia algorithms, such as video encoding, video decoding, and 3D graphics, place heavy demands on memory bandwidth and require low latency to sustain maximum processing throughput. To this end, Katmai provides several new capabilities, including microarchitectural enhancements to Pentium II's existing write-combining feature. The new improvements increase the sustainable bandwidth on a 100-MHz P6 bus by more than 20%. (Write-combining gathers stores to consecutive addresses into cache-line-sized chunks before bursting to the bus.)

Like RISC architectures dating back to at least the 88110 (see MPR 12/4/91, p. 1), KNI adds cache-prefetch instructions that give software a mechanism for bringing data into

the cache before the program actually needs it, thus allowing software to overlap processing with long-latency memory reads and to optimize cache and memory bus utilization. These prefetch instructions differ from normal loads in that they are just hints and never cause a program fault, making them less costly to issue than loads; for instance, they do not require a reorder-buffer entry.

The prefetch instructions can specify whether data will be prefetched into just the L1 cache, all cache levels, or all levels except L1. Data that will be used once would be brought into the L1 only. Data likely to be reused would be brought into all levels, unless it was not going to be used immediately, in which case it would be brought into the higher cache levels but not the L1.

KNI adds streaming-store instructions that store data directly to memory, bypassing the caches. This feature avoids throwing useful data out of the cache and polluting it with useless entries. These instructions also allow memory coherence to be managed explicitly by software, an advantage for streaming large data sets because bus-snooping overhead is eliminated. Intel says that proper use of KNI's prefetch and streaming stores can improve worst-case array-transfer rates by more than 2.5× on a 100-MHz P6 bus.

KNI also offers an SFENCE instruction that flushes the write-combining buffers. This instruction gives software special capabilities, such as ensuring that pixel updates get to the frame buffer and explicitly managing memory coherence.

FP Features	MMX w/KNI	MMX w/3DNow	MIPS V w/MDMX	PowerPC w/AltiVec
Registers	8 fp	8 int/fp	32 int/fp	32 int/fp
Vector Width	128 bits	64 bits	64 bits	128 bits
FP Precision	Single	Single	Single	Single
FP Parallelism	4 or 1	2	2	4
FP Arithmetic	IEEE & FTZ	FTZ	IEEE	IEEE & FTZ
Src Operands	2	2	3	3
FP Mul-Add	No	No	Yes	Yes (fused)
FP Min/Max	Yes	Yes	No	Yes
Cond Move	No	No	Yes	Yes
Estimates	1/x, $1/\sqrt{x}$	1/x, $1/\sqrt{x}$	No	1/x, $1/\sqrt{x}$, $\log_2$
Conversion	FP ⇔ Int	FP ⇔ Int	FP ⇔ Int	FP ⇔ Fixed
Data Reorg	Lmtd Swizzle	None	None	Permute 8Wd
Integer Features				
Registers	8 int	8 int/fp	32 int/fp	32 int/fp
Vector Width	64 bits	64 bits	64 bits	128 bits
Motion Est	$\Sigma \lvert a-b \rvert$	None	$\Sigma (a-b)^2$	$\Sigma (a-b)^2$
Motion Comp	Average	None	Average	Average
Speech	Min/Max	None	Min/Max	Min/Max
Data Reorg	Pack/Merge	Pack/Merge	Pack/Unpk	Permute 32By
Other Features				
Prefetch	Cache Line	Cache Line	None	4 Streams
Prefetch Ctl	L1,~~L1~~,All	None	None	Transient
Store Control	Cache Bypass	None	None	Stream Store
Write Gather	Yes, SFENCE	No	No	Yes, SYNC

Table 1. KNI adds significant new functions to MMX and is nearly a complete superset of 3DNow. (Source: vendors)

Living in an x86 World

Although KNI is a reasonably clean architectural extension, the constraints of the x86 architecture are evident. For example, eight registers, while better than none, are too few for many multimedia algorithms, especially with the register juggling required to circumvent the destructive nature of the x86's two-operand instruction format. Intel would likely have added more if not for the limitation of the 3-bit register field in the *modR/M* byte of the x86 instruction encoding. While register renaming relaxes the need for registers somewhat, it cannot substitute for a larger register namespace in dealing with the multitude of coefficients, transform matrices, and intermediate variables common in DSP algorithms.

As Table 1 shows, a feature notably absent from KNI (as well as from 3DNow) is multiply-add, the heart of vector dot product and the single most common operation in digital signal processing. Without multiply-add as a single instruction, Katmai will need nearly twice the instruction bandwidth in many critical DSP loops. Furthermore, with separate instructions, the add's dependence on the multiply's result serializes it behind a long-latency operation. Dynamic instruction reordering may fill some of the resulting pipeline bubbles, but it is not as effective as multiply-add. Software loop unrolling could help cover this latency, but KNI lacks sufficient register namespace to support much unrolling. Since a multiply-add unit is only slightly larger than a multiplier, we assume Intel omitted the feature to conform with the x86's two-operand format.

The scalar-form KNI instructions are mandated by a serious flaw in the x86/MMX architecture, which precluded efficient mixing of scalar FP and MMX instructions. Scalar-form instructions fix this problem—at least for single-precision operations—since they can be freely mixed with MMX instructions. The new scalar instructions also present a direct register-addressing scheme that floating-point programmers will welcome over the stack-based address model of the current scalar FP instructions. This feature comes, however, at the cost of introducing partial-register writes into the new KNI register file, an otherwise unnecessary complication.

Registers Solve and Create Software Problems

KNI's new registers add much-needed register namespace and width, and they solve MMX's problems with concurrent floating-point and MMX operation, but they come at a large software cost. The new registers must be saved and restored across context-switch boundaries, necessitating modifications to the operating system's first-level interrupt handlers and stack layout. Microsoft is undoubtedly not thrilled about this, but sources indicate that the company will accommodate the new state in Windows 98 and NT. All other x86 OS vendors (e.g., various Unix vendors) must also make this change if they wish to support KNI.

The new state raises other software obstacles as well. Compilers must be modified for procedure calling and parameter passing. A protocol must be defined to allow KNI pro-

grams to call old libraries that were compiled without knowledge of the new state. And applications and libraries, like DirectX, must be updated to take advantage of KNI. Intel has been working with software vendors for some time to pave the way, but the new state will make KNI's adoption more painful than MMX's or 3DNow's.

KNI KO's 3DNow

Perhaps Intel would have been further ahead if the original MMX had included KNI's features. Had MMX included new registers, for example, the problem of interference with scalar floating-point operations could have been avoided from the start. In addition, integer MMX operations would have benefited from the 128-bit parallelism, and the OS and compiler issues would now be behind it. The ISA extension would also have been cleaner if both integer and floating-point SIMD operations used the same register file. Furthermore, Intel would have preempted 3DNow, avoiding this loss of control over the x86 architecture.

On the other hand, had MMX included KNI's additional state, software vendors may have been more reluctant to adopt MMX and SIMD processing in general. So even though the final architecture may be less elegant, in retrospect Intel's strategy may have been the right one.

Many details of KNI will not be revealed until next year, although Intel is expected to provide more information on Katmai's implementation of it at Microprocessor Forum next week. But from an architectural perspective, KNI appears to be about as clean an architectural extension as is possible within the constraints of the existing x86 and MMX architectures. Although KNI is not as comprehensive as PowerPC's AltiVec, as Table 1 shows, it clearly improves on MMX and is far more powerful than AMD's 3DNow, offering new registers and twice the architectural parallelism.

KNI puts the future of 3DNow in serious jeopardy. Although 3DNow has a software lead, the two extensions are clearly redundant, and KNI is more powerful. With the enormous investment Intel is pouring into software and development tools, and with Katmai deployments going into full swing next year, 3DNow will be hard pressed to hold the attention of software developers . For AMD, Cyrix, and IDT, the best course of action may be to read the handwriting on the wall and modify their next-generation processors to support the KNI architecture. M