MICROPROCESSOR REPORT
THE INSIDERS' GUIDE TO MICROPROCESSOR HARDWARE
VOLUME 9 NUMBER 10 JULY 31, 1995

P6 Underperforms on 16-Bit Software

Windows 95 Needed to Outrun Pentium; NT Does Even Better

by Linley Gwennap

Causing some PC users to rethink their upgrade plans, Intel revealed test data that shows its forthcoming P6 processor is tuned for 32-bit software, leaving performance on 16-bit programs below that of a fast Pentium. Because Windows 3.1 and its applications run in 16-bit mode, nearly all current PC software will show little if any performance gain when executed on a P6. In many cases, initial P6 systems will be slower than high-end Pentium PCs.

Intel is banking on a quick shift to 32-bit software. Computer Intelligence InfoCorp expects Microsoft's 32-bit Windows 95 operating system, due to ship within a month, to sell more than 60 million copies in its first year of release. This volume potential has most major software vendors planning to deliver 32-bit versions of their applications within that year. While the P6 performs well on Windows 95, it is hampered by the significant amount of 16-bit code that remains in that OS; Windows NT, a pure 32-bit operating system, is required to maximize P6 performance.

The P6 will enter the market later this year in high-end desktops and servers. Many of these systems will use existing 32-bit operating systems such as NT, Novell NetWare, and various versions of Unix; the rest will ship with Windows 95. P6 volume will continue to be relatively low and restricted to high-end systems throughout 1996. Well before the P6 enters the PC mainstream in 1997, 32-bit PC applications will be widely available.

We expect that the processor's 16-bit performance problems will have relatively little effect on P6 sales. But the design trade-offs that inhibit 16-bit performance will cause problems for PC buyers who want higher performance but don't want to buy new 32-bit software to get it. Through 1996, these users may delay buying P6 systems or turn to Intel's competitors, which have placed more emphasis on 16-bit performance. Most users, however, are likely to be satisfied by high-end Pentium systems in this timeframe.

Intel Blames Baroque Code

In this context, the phrase "16-bit code" is a shorthand for older code that makes heavy use of certain features of the x86 architecture, specifically segment writes, partial register operations, and unaligned data accesses. Some recent x86 software, even though written to a 16-bit model, avoids many of these features and will run well on a P6. But most current software contains older code that was written directly in x86 assembly language and/or hand-optimized to improve performance. The P6 does not efficiently execute older code that frequently uses these features.

Although previous Intel processors have been able to perform these functions with little or no penalty, they cause severe problems in the decoupled architecture of Intel's newest processor. The P6 includes register renaming, speculative execution, and a deep pipeline *(see **090202.PDF**)*, all of which make it more difficult to efficiently execute these functions.

Specifically, a write to a segment register cripples the P6's speculative execution. Segment writes cannot be executed speculatively in the P6; all previous instructions must be drained from the pipeline before a segment write can occur. Furthermore, because a segment change can affect the execution of all subsequent operations, instructions following the segment write cannot be executed out of order but must wait for the segment register to be updated.

The P6 typically has many instructions executing speculatively and out of order at any given time. A segment write forms a barrier that restricts instruction reordering, severely reducing throughput. This serialization of instructions also occurs on a mispredicted branch; a segment write, however, requires 10–15 cycles of microcode on top of the serialization effects, resulting in a total cost of 20–30 clocks.

Unfortunately, segment writes are common in existing code because the 16-bit addressing model limits the amount of memory per segment to 64K. Accessing larger data structures or code sequences requires changing segments. In addition, programmers frustrated by the limited register set of the 8086 often used the segment registers for temporary storage. The 32-bit model increases the segment size to 4G, eliminating the need to change segments in most applications, and discourages coders from storing temporary values in segment registers.

Segment writes are not a significant problem for Pentium because it does not execute instructions speculatively or out of order. In addition, Pentium sports a segment-descriptor cache that reduces the execution time of a segment write to just a few cycles on hits to that cache. The P6 does not have such a cache.

More Stupid Programming Tricks

The complex x86 register model causes problems for the P6's register renaming. Modern x86 processors provide eight 32-bit registers (EAX, etc.) but, to retain compatibility with code written for the 8086 or 80286, allow some of these registers to be accessed as 16- or 8-bit quantities as well. For example, the lower 16 bits of EAX can be addressed as AX, which is subdivided into two 8-bit values, AH and AL.

An x86 processor like Pentium handles this model easily, pulling the requested number of bits from the register file on each access. The P6, however, typically takes operands from the reorder buffer (ROB) rather than the register file. If a program performs a series of 8-bit calculations on AL, for example, the P6's ROB works well. But if the program then reads from AX, problems arise.

In this situation, half of AX (namely AL) exists in the reorder buffer, but the other half may be somewhere else in the ROB or may be in the physical register file. To handle this situation cleanly, the P6 would have to allow each operand to consist of two possibly separate halves, doubling the number of buses, register tags, and other logic. Instead, the P6 stalls the offending instruction until all predecessors complete; at that point, the full value can be obtained from the register file.

Code generated by compilers typically uses a consistent data width and thus does not pose this problem to the P6. Assembly-level programmers, however, often use this trick to insert data into a larger value, saving a cycle here or there. Now this obscure coding practice is coming back to haunt Intel.

Unaligned data is a third area that causes delays in the P6. In a simple pipelined processor like Pentium, unaligned data can be handled by a minimal amount of hardware, adding a one-cycle delay. On the P6, a misaligned store takes only one extra cycle (which is typically masked by the store buffer), but a misaligned load costs about seven cycles.

For all loads, the P6 must check the load address against the addresses in the store buffer in case the data in the store buffer is more up-to-date than that in memory. Because a misaligned load can span two cache lines, two different aligned load addresses would have to be checked at the same time. Instead, the P6 holds off the load (and any dependent instructions) from executing until all preceding instructions complete, causing the lengthy penalty.

In early x86 processors, data alignment was not an issue, so programmers did not enforce alignment; in fact, they often ignored it to save space. More recent compilers usually align data to avoid the delays in Pentium. Also, many software vendors have been eliminating unaligned data to simplify porting their code to RISC processors, most of which strictly enforce alignment. Although the 32-bit x86 coding model still allows unaligned data, Intel expects few ISVs to take advantage of this feature in 32-bit applications.

Ironically, the code constructs that Intel now bemoans exemplify the CISC baggage that RISC vendors rejected when creating their architectures. All desktop RISCs support a full 32-bit model with large segments and no partial register accesses. Only PowerPC allows unaligned data accesses. It appears that, to achieve high performance, Intel's latest x86 processor requires RISC-like code streams, penalizing PC software that takes full advantage of the original x86 feature set.

AMD, Cyrix Avoid 16-Bit Problems

Both AMD and Cyrix claim that their latest processors, the K5 and M1, perform well on both 16-bit and 32-bit code. These chips are better able to handle the thorny issues noted above, in part because they have simpler designs than the P6, aiming for lower clock speeds and performance. In addition, the two Intel competitors placed more emphasis on maintaining high performance on existing code.

The M1, for example, has additional circuitry, to speed the handling of segment register writes. Cyrix declined to describe the specifics of this circuitry due to pending patent applications, but unlike the P6, the M1 does not stall for most writes to the segment registers.

Cyrix's chip implements register renaming and has a problem similar to the P6's in handling operand reads when only part of the register has been recently updated. Like the P6, the M1 waits for the partial register update to be complete before proceeding. But the M1 allows less out-of-order execution than the P6, so the impact of this problem is far less than in the Intel chip. The Cyrix design handles unaligned data reads with a single penalty cycle, much more quickly than the P6.

The K5 microarchitecture is similar to the P6's in its decoupled RISC-like design, yet AMD wanted to ensure strong performance on older code. The company included extra tag fields and comparators in the reorder buffer to handle partial register accesses smoothly. In cases where the P6 must stall, the K5 can assemble the needed value from multiple sources, incurring only a slight penalty. AMD claims that this feature adds only two 4-bit tag fields to each ROB entry and requires several 4-bit comparators, a minor impact.

The K5 can execute segment changes speculatively. For example, a change to the data segment is issued to the load/store unit. Any subsequent loads or stores are dependent on that operation and cannot be committed before the segment change. This technique, which the P6 designers chose not to implement, avoids significant performance penalties. The K5 also handles unaligned data accesses with a single penalty cycle.

Clock Speed Looking Good

Intel revealed that the initial P6 parts are running at higher clock speeds than originally anticipated. In addition to 133-MHz parts, the company now expects to deploy 150-MHz chips from the same 0.5-micron BiCMOS process. This would indicate that a quick shrink to Intel's 0.35-micron BiCMOS process should easily push clock speeds to 200 MHz in 1H96. A redesign to the upcoming 0.28-micron CMOS process *(see* ***090905.PDF****)* could result in clock speeds of up to 266 MHz by the end of 1996.

Figure 1 shows Intel data comparing a 133-MHz Pentium system with a hypothetical 150-MHz P6 system. The P6 results are extrapolated from first silicon running at 133 MHz. Both systems use 256K of secondary cache; the Pentium system has synchronous SRAM, while the P6 relies on its built-in L2 cache.

The Pentium board features the latest Triton chip set, and the P6 uses the Orion PCIset *(see* ***090701.PDF****)*. Both use the same memory and graphics subsystems and the same software. The P6 is slightly hampered by a chip-set bug that prevents the use of a bus-mastering IDE interface. Performance is shown relative to a 100-MHz Pentium in the same system configuration as the 133-MHz Pentium.

On Windows 3.1 applications, represented here by SYSmark95, Pentium outperforms the P6 by 16%. Moving to Windows 95 allows the P6 to pass Pentium in performance, but by just 25%. Only on full 32-bit operating systems, such as Windows NT or Unix, does the P6 reach its full potential, outrunning the high-end Pentium box by about 50%. Note that, because the SPECint92 benchmark is cache-bound, SYSmark95 is a better measure of performance for most 32-bit applications.

Table 1 (see below) provides a more detailed comparison between these two systems. On 16-bit application tests run under Windows 3.1, the 133-MHz Pentium outperforms the 150-MHz P6 by 2–15%. But on 32-bit programs under Windows NT, the P6 beats the Pentium system by up to 74%. The biggest gap is for programs that heavily use floating-point math, where the P6 has an advantage over Pentium; integer applications cluster around the 54% advantage shown on SysMark NT.

Price & Availability

Intel has not yet announced the P6 processor or the Orion PCIset; systems are expected in 4Q95. For more information, contact your local Intel sales office or check the World Wide Web at *http://www.intel.com/procs/p6*.

Intel has released an estimate of 200 SPECfp92 for the 133-MHz P6, exactly what we predicted when the part was first revealed *(see* ***090201.PDF****)*. Intel expects the 150-MHz version to achieve at least 220 SPECint92 and 215 SPECfp92. The faster P6 has a 116% advantage on the SPECfp92 benchmark compared with the 133-MHz Pentium system described previously.

Careful Positioning Required

With the disclosure of this performance data, Intel has begun to carefully position the P6. Although Pentium has been marketed as faster and better for all PC users, the P6 requires a slightly different spin. For new PC users, the problem is moot: by buying Windows 95 and a 32-bit version of Microsoft Office with their P6 systems, these users are pretty much covered.

Most PCs, however, are sold to business or home users who have already invested in at least some PC software. These users may want to maintain their investment by continuing to use this software on a new system. Unfortunately, if the software runs under Windows 3.1 and the new system contains a P6 processor, performance will be disappointing to users expecting to see a big improvement from the P6.

At any other time, this desire to reuse software

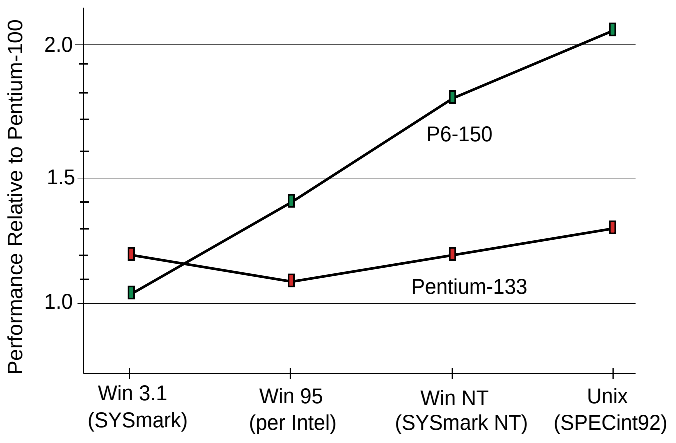

Figure 1. The P6 is slower than a high-end Pentium on the 16-bit applications used with Windows 3.1 but excels on pure 32-bit code under Windows NT or Unix. See text for configuration details. (Source: Intel)

		Pentium-133 relative to Pentium-100	P6-150 relative to Pentium-100	P6-150 relative to Pentium-133
Win 3.1	Lotus 123 5.0	1.20	1.10	0.92
	Paradox 5.0	1.12	1.10	0.98
	PowerPoint 4.0	1.20	1.05	0.88
	Word 6.0	1.18	1.00	0.85
	SYSmark95	1.17	0.99	0.84
Win 95 (beta)	Excel	1.16	1.42	1.22
	Freelance	1.32	1.63	1.23
	PowerPoint	1.28	1.70	1.32
	Aggregate	1.26	1.58	1.25
Win NT	MathCAD	1.24	2.16	1.74
	Photoshop	1.25	1.69	1.35
	Typestry	1.27	1.91	1.50
	Vista Pro	1.25	1.69	1.35
	SYSmark NT	1.20	1.85	1.54

Table 1. These performance ratios show that a 133-MHz Pentium system can outrun a 150-MHz P6 on 16-bit applications, but the P6 shines on 32-bit code. See text for configurations. (Source: Intel)

would be a major stumbling block for the P6. Windows 95, however, represents the biggest functional change in Microsoft's OS line since Windows 3.1 debuted in 1991. We expect users to flock to the new OS, and most of these users will want new applications that take advantage of the features of Windows 95. Software vendors are likely to offer low-cost upgrades for users who already own 16-bit applications. In short, we expect that most PC buyers will want to upgrade their software.

Only performance-critical software is affected by the move to the P6. The forthcoming processor retains full x86 compatibility and will run any software package, 16- or 32-bit. Many utility programs and older applications require little CPU performance and perform adequately on P6 systems. Users need to focus on programs that can benefit from more efficient CPU usage.

For PC buyers sticking with Windows 3.1 and 16-bit applications, a Pentium-based system will provide similar or better performance than initial P6 systems and will do so at a much lower price. The P6 system will provide better performance over time as these users move to 32-bit code. On Windows 95, the P6 has a performance advantage over Pentium, but not enough to justify a significant price premium. For users of Windows NT, the P6 is great. In short, a Pentium system provides the best value today, while a P6 system has more headroom for future software.

As the P6 reaches 200 MHz and beyond in 1996, these positioning problems become easier to solve. A 200-MHz P6 should best the fastest Pentium on virtually any program, 16- or 32-bit. For early P6 parts, however, the performance shortfall on Windows 95 programs may cause Intel to price these parts more aggressively than previously expected.

A Competitive Opportunity

The P6's lack of 16-bit performance offers an opening to other x86 processor vendors, but none is likely to take advantage of it. We do not expect Cyrix's M1 or AMD's K5 to significantly exceed the performance of the fastest Pentium processors until early 1997 *(see **090804.PDF**)*. At that time, these alternative processors are likely to be the fastest x86 processors available for 16-bit code; that is, they should outperform the P6 in that mode. By 1997, however, few performance-focused users will still be running 16-bit code.

If AMD or Cyrix could deliver super-Pentium performance in 1996, they would benefit from the P6's problems, but we do not expect this to occur. NexGen plans to deliver its 686 next year; if this chip exceeds Pentium's performance on 16-bit code, it could take some sales from the P6. On the other hand, the 686 may have the same 16-bit performance problems as the P6.

The upcoming software transition also leaves an opportunity for PowerPC or other RISC chips, but again, none seems likely to exploit it. IBM, for example, could argue, "If you're going to switch to Windows 95 and throw away your old 16-bit software, why not switch to NT on PowerPC instead and buy new RISC software?" Microsoft plans to add the Windows 95 user interface to NT next year, making this argument even more compelling.

The flaw here is that the P6 delivers Pentium-class performance on 16-bit x86 applications, while no RISC processor can make that claim. Users moving to the P6 can keep at least some of their old x86 software, but a move to RISC requires virtually all new software. If a RISC vendor could deliver a processor with fast x86 emulation within the next year, it could offer a convincing reason to switch from a Pentium PC, but we don't foresee such a device in that timeframe.

Intel Poised to Win Gamble

In 1992, when Intel was putting together the P6 design goals, Windows 95 (then known as Chicago) was supposed to provide a fully 32-bit environment and ship in 1994, well before the P6. Still, it is surprising that Intel was willing to risk its x86 empire by forsaking its biggest strength, the huge installed base of x86 applications. The company argues that adding features to speed 16-bit code would have either lowered the P6's performance on 32-bit code or increased its cost. Adding features also would have increased schedule risk.

Today, we see Windows 95 barely beating P6 systems to the wire. Worse yet, the final version includes large chunks of 16-bit code, left there to help Microsoft meet its memory-size goals without further jeopardizing its own schedule. This result leaves the P6 with a performance shortfall, although with no competitor poised to take advantage of it, the impact on P6 sales in 1996 should be small; a few buyers may delay purchases and stick with Pentium for a bit longer.

The critical time for the P6 is 1997, when we expect the processor to plunge into the PC mainstream, much as Pentium is doing this year. At that time, Windows 95 (or its successor) will still be the mainstream operating system, although Windows NT should be gaining share *(see* ***0910ED.PDF****)*. AMD and Cyrix will be weighing in with next-generation parts that could better exploit the P6's weakness on 16-bit code.

If Microsoft modifies Windows 95 over time to reduce the amount of 16-bit code, it would help Intel ramp P6 sales in 1997. Alternatively, the P6 would benefit from a faster transition to Windows NT. But Intel has been burned by Microsoft once already. To ensure the P6's success in the mainstream, Intel could modify the core design to fix the problems noted above. Having now realized the problem, the company has plenty of time to ready an enhanced part for 1997 shipments. Such enhancements would be a worthwhile insurance policy against continued reliance on 16-bit code by the masses.

While recoding and recompilation have often been required to achieve optimal performance on a new Intel processor, never before have some programs actually slowed down when moved to the next-generation CPU. The P6 performance profile sets the stage for a P7 that, at least in one incarnation, implements a new VLIW-like instruction set but retains compatibility with existing x86 software. This chip may very well be slower than a high-end (e.g., 300-MHz) P6 on unmodified software, but it should offer a significant speedup on recompiled code. If users accept this premise for the P6, it bodes well for an eventual transition to a completely new instruction set. ♦