MICROPROCESSOR REPORT

THE INSIDERS' GUIDE TO MICROPROCESSOR HARDWARE

VOLUME 6 NUMBER 12 SEPTEMBER 16, 1992

Intel Begins Gradual P5 Unveiling

Many Details Still Missing—Most Questions Go Unanswered

By Michael Slater

At last month's Hot Chips conference, Intel let out the first concrete information about the P5. The presentations were highly constrained; many features of the chip were not described at all, and even for those areas that were described, Intel declined to provide details or respond to most questions. Each presentation (one on the integer unit, and one on the floating-point unit) was begun with a legal disclaimer that the material may be the subject of pending patents, which met with widespread hissing from the audience. Most questions were answered by saying, essentially, "that's a good question, and we did exactly the right thing."

There was no information, of course, regarding price or availability, and only very general statements on performance. Among the other issues declared "off limits" were cache sizes, details of the cache organization, bus interfaces, architecture extensions, and instruction cycle counts. Nevertheless, compared to the paucity of information previously released on the P5, a significant amount of new information was revealed.

While Intel will not discuss pinouts or bus interfaces, it is rumored that P5 will be offered in both 32- and 64-bit external bus configurations. One version, codenamed P24T, will use an extended OverDrive socket pinout and will be marketed as an upgrade processor for 486DX2 systems.

Integer Unit

The P5 is a two-issue superscalar machine. The instruction issue approach is to decode two instructions in each clock cycle, and issue both of them if:

- Both are "simple" instructions.
- The first instruction is not a jump.
- The second instruction does not use the results of the first.
- The two instructions do not use the same destination.

If these conditions are not met, then the first instruction of the pair is issued alone. In the following cycle, the second instruction of the pair is coupled with the next sequential instruction for possible dual-issue. Instructions are never issued out of order. "Simple" instructions were described as generally including ALU operations, move instructions, and jumps. "Simple" ALU and move instructions include not only the RISC-like register-to-register and immediate-to-register forms, but also memory-to-register and register-to-memory formats.

The pipeline design, as shown in Figure 1, follows the same general structure as the 486, with five stages: fetch, decode1, decode2, execute, and write-back. The first two stages process a pair of instructions. The last three stages are duplicated, forming two separate pipelines, called the U-pipe and the V-pipe. When the conditions described above for dual-issue are met, instructions are issued to both pipes; otherwise, only the U-pipe is used. Each pipe has a full integer ALU, allowing two integer instructions to be executed in each clock cycle (when there are no dependencies).

Because the x86 architecture has a relatively small register set, as well as instructions that combine memory references with computations, the number of data memory references per instruction is considerably higher than for RISCs. Intel estimates that optimized, 32-bit x86 code has an average of 0.6 data references per instruction, while standard RISCs average about 0.3 data references per instruction. Because of this high frequency of data memory accesses, it was important for the P5 to allow two such references to occur simultaneously.

The P5 makes this possible by using a dual-access data cache, as shown in Figure 2. The TLB (translation look-aside buffer) and the cache tags are fully dual-ported. The cache data array itself is single-ported, but it is divided into multiple, interleaved banks. Two separate accesses can be performed in the same cycle as long as they are to separate banks. If a bank conflict occurs, the Vpipe access is stalled for one cycle. Don Alpert, a P5 architect who gave the integer unit presentation, said

 © 1992 MicroDesign Resources

that a fully dual-ported cache was considered, but the denser, single-ported cache structure allowed the cache to be larger, and the resulting increase in hit rate more than compensated for the loss in performance due to stalls resulting from bank conflicts.

Intel admitted that there were separate instruction and data caches and that the data cache was optionally write-back, but the cache associativity, cache size, and other organizational details were not disclosed. Coherency between the instruction and data caches is provided, so self-modifying code will continue to work, but the mechanism that implements this was not described.

It has been rumored that the P5's instruction cache includes additional bits for predecoded opcode information and expanded, aligned fields. Intel declined to respond to questions about this possibility.

A branch target buffer (also called a branch history table) caches the destination address for previously encountered branches, along with additional bits of history information indicating how frequently each branch has been taken. (Intel would not disclose the size of this cache, the number of bits of history data, or the penalty for mispredicted branches.) This allows correctly predicted branches to execute with no pipeline delays. Note that this is not a branch target cache as implemented in 29000; the P5's branch target buffer stores only addresses, whereas the 29000-type branch target cache stores the first few instructions at each branch destination.

Floating-Point Unit

In the past, the floating-point performance of x86 microprocessors has been poor. Even with the 486, the SPECfp92 rating is less than half the SPECint92 rating. This is not primarily a result of the x86 architecture, but rather of Intel's priorities: making floating-point go fast takes lots of transistors, and in traditional PC markets it isn't that important. Thus, Intel did not devote much design effort or transistor budget to the floating-point unit in the 486.

With the P5, however, all that has changed. While the floating-point needs of the typical PC user haven't increased much, it has become strategically important for Intel to match the performance of RISC microprocessors, whose biggest performance lead is in floating-point. PC applications are becoming more floating-point-intensive with increased use of 3-D graphics, and Intel also hopes to push the P5 into technical workstation markets where fast floating-point is essential.

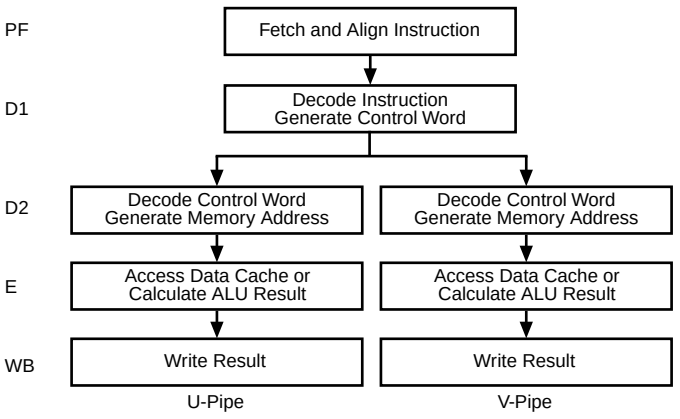

Figure 1. P5 integer pipelines.

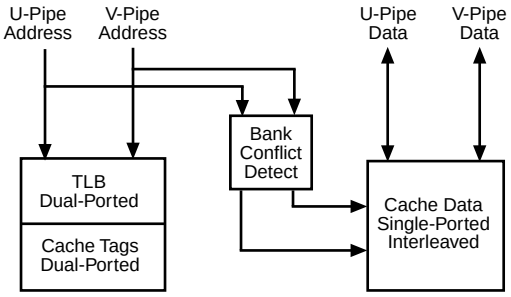

Figure 2. Dual-access data cache in the P5.

The P5's floating-point unit is fully compatible with that in the 486. There have been rumors of a new floating point model that would supplement the register stack with a traditional register file, but it appears that if this is done at all it will be in the P6, not the P5.

The eight-stage floating-point pipeline is integrated with the integer pipelines, and the first four stages are the same. Both the U-pipe and the V-pipe are used to fetch operands, allowing both data cache access paths to be used in parallel to load a 64-bit floating-point value in a single clock cycle. Floating-point execution is performed in the U-pipe.

The integer execute stage is used to fetch operands, and it is followed by three floating-point execution stages. The final stage of the floating-point pipeline is used for error reporting; results of calculations are available at the start of this stage, so it does not affect latency.

Like most high-end RISCs, the P5's floating-point unit is fully pipelined for add/subtract and multiply operations; it can start a new operation on every clock cycle, assuming no dependencies. This rate can be maintained for double-precision, memory-to-register operations (assuming a cache hit, of course) and also for extended precision (80-bit), register-to-register operations.

Some architectures, such as DEC's Alpha, sacrifice precise exceptions to improve floating-point performance. This means that one or more instructions beyond the instruction causing the exception may be executed before the exception is recognized. Intel did not have this option if full compatibility with existing programs was to be maintained, but having to wait until a floating-point instruction was complete before launching the next instruction would cause a significant loss in performance.

The P5 tackles this problem by adding hardware

 © 1992 MicroDesign Resources

that examines the input operands for each floating-point operation that could generate an exception to determine if the calculation is "safe"; that is, if it can be guaranteed not to generate an exception. Only if the operation has the potential to generate an exception is the next instruction delayed until the first operation completes. According to Intel's P5 floating-point design manager Dror Avnon, who gave the presentation, unsafe operand combinations are very rare. Exceptions in the P5 are rarer than in typical double-precision floating-point units because all data is stored in the register stack in the 80-bit extended-precision format, providing additional bits in the exponent. Avnon said that in the entire SPECfp89 suite, no unsafe operations were detected.

The floating-point adder and multiplier provide single-cycle throughput and three-cycle latency for all precisions (single, double, and extended). Intel did not provide throughput or latency figures for the more complex operations, but Avnon did say that the divider processes two bits of quotient in each cycle. (For a double precision floating-point number, which has a 52-bit fraction, this implies a divide time of 26 cycles plus a few more for setup and normalization.)

The P5's floating-point unit is the first high-performance design to implement transcendental functions. These functions aren't included in RISC instruction sets; Motorola decided to trap these operations in the 68040 and implement them using trap handlers. The P5 abandons the Cordic algorithms used by the 486's FPU and earlier x87 coprocessors, and instead uses table-driven algorithms with polynomial approximation. Both pipelines are used to implement these algorithms. The resulting performance was stated as 4 to 6 times a 486DX-33, which means that the per-clock performance improvement is a factor of 2 to 3.

In general, floating-point instructions cannot be dual-issued because each instruction uses both pipes. There is at least one exception, however; the FXCH instruction, which exchanges the top-of-stack with a register deeper in the stack, can be issued and executed in parallel with a floating-point computation instruction. This is important because the top-of-stack serves as the floating-point accumulator, creating a bottleneck from which register-file-oriented floating-point processors don't suffer, and the parallel execution of the exchange instruction eliminates much of the bottleneck. The exchange is performed after the computation completes, so it has the effect of directing the result to any register in the stack. At the same time, it brings a value up from that register into the top-of-stack, where it can be used by the next instruction.

Compiler Issues

As with most superscalar processors, extracting the full performance of which the hardware is capable requires a compiler that properly optimizes for the processor's pipeline structure. The usual techniques of instruction scheduling, register allocation, and loop unrolling all apply to the P5. Good register allocation is especially important, since the register set is relatively small. In addition, there are some considerations that differ from those for RISC processors. For example, it is important that the compiler select "simple" opcodes whenever possible, since only these instructions can be dual-issued. For floating-point code, different code-generation strategies are required to take advantage of the ability to parallel-issue the exchange instruction with computation instructions.

In the PC world, where there is a massive installed base of existing applications, the ability to perform well on old binaries is important. It remains to be seen how much of the P5's potential performance boost will be realized on old binaries. The most performance-critical programs, however, are likely to be recompiled relatively soon, and the P5-specific optimizations are expected not to hurt performance on earlier processors. Intel has been working with its own compiler group and with outside compiler vendors in an effort to have P5-specific compilers available simultaneously with the release of the processor.

Performance

At its target clock rate of 66 MHz, Intel claims that performance will be "over 100 MIPS," where MIPS are defined as Dhrystone 1.1 performance relative to a VAX 11/780. In other words, Dhrystone performance will be higher than 175.7 KDhrystones/s. For comparison, a 486DX2-66 is rated at 94.8 KDhrystones/s, giving the P5 a 1.85 times speedup over that processor. A 486DX-50 is rated at 72.8 KDhrystones/s, so the P5 is about 2.4 times faster than that chip on Dhrystone. Intel's "100 MIPS" rating assumes that Dhrystone is recompiled with a P5-optimized compiler, so the speedup of existing binary programs will be somewhat less.

Intel did not give any information on instruction clock cycle counts, nor did it provide any information on possible speedups in some of the more complex, slower executing integer instructions—ones that aren't "simple," and can't be dual-issued.

For floating-point performance, Intel claimed a speedup of 4 to 6 times for scalar code and 6 to 10 times for vectorizable code. The 4 to 6 times speedup is claimed to be achieved even on existing 486 binaries. However, these ratios are relative to a 33-MHz 486, so they must be cut in half to compare with the high-end 486DX2-66. Taking this into account, it appears that typical 486 floating point programs will be 2 to 3 times faster on a P5-66 than on a 486DX2-66. The P5 is also expected to allow a full speed bus interface (i.e., not at half the processor speed, as in the DX2), which would further improve performance on programs with poor cache performance.

Intel hasn't released any projected SPEC ratings,

 © 1992 MicroDesign Resources

but we can make some guesses. If the improvement in SPECint is proportional to the improvement in Dhrystone (and that is a big if), then the SPECint92 rating of the P5-66 could be as high as 72 (2.4 times the 486DX-50 rating of 30.1). Assuming that Dhrystone overstates the chip's performance, as it typically does, a SPECint92 rating in the 60s seems likely.

Estimating the P5's SPECfp92 performance is difficult, since it is hard to know where in the speedup range of "4 to 10 times a 486DX-33" each benchmark will fall. Furthermore, Intel has provided SPEC92 results only for the 50-MHz 486. To take a wild guess, if we assume that the average speedup over a 486DX-33 on the SPEC floating-point suite is a factor of 6, this would be about 4 times the 486DX-50, which has a SPECfp92 rating of 14.0. This implies a SPECfp92 rating of about 56 for the P5-66. Note that this could be way off (in either direction), since we have just guessed where the average speedup on the SPEC suite is within Intel's projected range of 4 to 10.

For comparison, Sun's SPARCstation 10/41, using a 40-MHz SuperSPARC, is rated at 52.6 SPECint92 and 64.7 SPECfp92. The fastest MIPS machine is SGI's Crimson, using a 50/100-MHz R4000SC, which is rated at 61.7 SPECint92 and 63.4 SPECfp92. While making detailed comparisons based on the above guesswork extrapolations from Intel's scant data would be foolish, it seems that the P5 is likely to be in the same ballpark as these high-end RISCs. It should be noted, however, that it will begin production perhaps a year after the R4000 and six months after SuperSPARC, and faster versions of both RISCs are likely to be shipping by the time P5 systems are in volume production.

Conclusions

Intel appears to have found ways to overcome many of the x86's architectural handicaps. The small register set, for example, means that there are more memory references, but the dual-access data cache helps minimize the performance impact. The stack-oriented floating-point register file creates an accumulator bottleneck, but the parallel execution of the exchange instruction reduces its effect. In this sense, the P5 appears to support the contention that the x86's architectural handicaps can be overcome with some implementation creativity.

On the other hand, the P5 also shows how the complexity of the architecture increases Intel's design tasks. For example, RISC processors have not had to go to the complexity of dual-access data caches to reach comparable performance levels. This also illustrates how the x86's architectural limitations affect many aspects of the design; it is not as simple as designing a nice, clean RISC processor with a small "compatibility unit" on the side, as some proponents have described the P5. While the P5 may achieve the same performance as the R4000, it will do so a year later, with three times as many transistors, and with a more complex process technology (BiCMOS).

At first glance, the P5 may appear to be not as aggressive in its issue strategy as some of the latest RISCs. It is limited to two instructions per clock, while SuperSPARC and IBM's RS/6000 can issue three instructions under optimal circumstances. The P5 cannot issue integer and floating-point operations in the same cycle, as can all superscalar RISCs.

The greater semantic content of x86 instructions, however, means that two instructions often do the same work that would require three or more instructions in a RISC architecture. For example, the memory-to-register instructions in the x86 architecture eliminate the need for separate load and store instructions. This also makes it less important to issue floating-point and integer instructions together, since many of the integer instructions in floating-point programs are loads and stores. Address calculation instructions are also sometimes eliminated by the x86's richer addressing modes. The P5's dual integer units are matched only by SuperSPARC and Motorola's 88110; other superscalar designs can issue only one integer computation instruction per cycle.

Until full details and measured performance data for the P5 are released, a complete evaluation is impossible. Rumors persist that the P5 has serious bugs, that the current silicon cannot reach the 66 MHz target clock rate, and that power dissipation is very high. While the design is surely impressive and will take the x86 architecture to new heights, it remains to be seen how it will compare to other processors shipping in the same timeframe. ♦

The P5 striptease will continue at the Microprocessor Forum on October 14.

 © 1992 MicroDesign Resources