THE HP PA-8000 RISC CPU

Ashok Kumar

Hewlett-Packard

This aggressive, four-way, superscalar microprocessor combines speculative execution with on-the-fly instruction reordering.

The PA-8000 RISC CPU is the first of a new generation of Hewlett-Packard microprocessors. Designed for high-end systems, it is among the world's most powerful and fastest microprocessors. It features an aggressive, four-way, superscalar implementation, combining speculative execution with on-the-fly instruction reordering. The heart of the machine, the instruction reorder buffer, provides out-of-order execution capability.

Our primary design objective for the PA-8000 was to attain industry-leading performance in a broad range of applications. In addition, we wanted to provide full support for 64-bit applications. To make the PA-8000 truly useful, we needed to ensure that the processor would not only achieve high benchmark performance but would sustain such performance in large, real-world applications. To achieve this goal, we designed large, external primary caches with the ability to hide memory latency in hardware. We also implemented dynamic instruction reordering in hardware to maximize instruction-level parallelism available to the execution units.

The PA-8000 connects to a high-bandwidth Runway system bus, a 768-Mbyte/s split-transaction bus that allows each processor to generate multiple outstanding memory requests. The processor also provides glueless support for up to four-way multiprocessing via the Runway bus. The PA-8000 implements the new PA (Precision Architecture) 2.0, a binary-compatible extension of the previous PA-RISC architecture. All previous code executes on the PA-8000 without recompilation or translation.

Architecture enhancements

PA 2.0 incorporates a number of advanced microarchitectural enhancements, most supporting 64-bit computing. We widened the integer registers and functional units, including the shift/merge unit, to 64 bits. PA 2.0 supports flat virtual addressing up to 64 bits, as well as physical addresses greater than 32 bits (40 bits on the PA-8000). A new mode bit governs address formation, creating increased flexibility. In 32-bit addressing mode, the processor can take advantage of 64-bit computing instructions for faster throughput. In 64-bit addressing mode, 32-bit instructions and conditions are available for backward compatibility.

In addition, the following extensions help optimize performance for virtual memory and cache management, branching, and floating-point operations:

- fast TLB (translation look-aside buffer) insertion instructions,
- load and store instructions with 16-bit displacement,
- memory prefetch instructions,
- support for variable-size pages,
- halfword instructions for multimedia support,
- branches with 22-bit displacements,
- branches with short pointers,
- branch prediction hinting,
- floating-point multiply-and-accumulate instructions, and
- multiple floating-point compare-result bits.

Key hardware features

Since the PA-8000's completely redesigned core uses no circuitry from previous-generation processors, we could design the new processor with any microarchitectural features necessary to attain high performance. Figure 1 (next page) shows a functional-block diagram of the PA-8000's basic control and data paths.

The chip's most notable feature is the 56-entry instruction reorder buffer, to our knowledge the industry's largest, which serves as the central control unit. It supports full register renaming for all instructions in the buffer and tracks instruction interdependencies to allow dataflow execution through the entire 56-instruction window.

0272-1732/97/$10.00 © 1997 IEEE

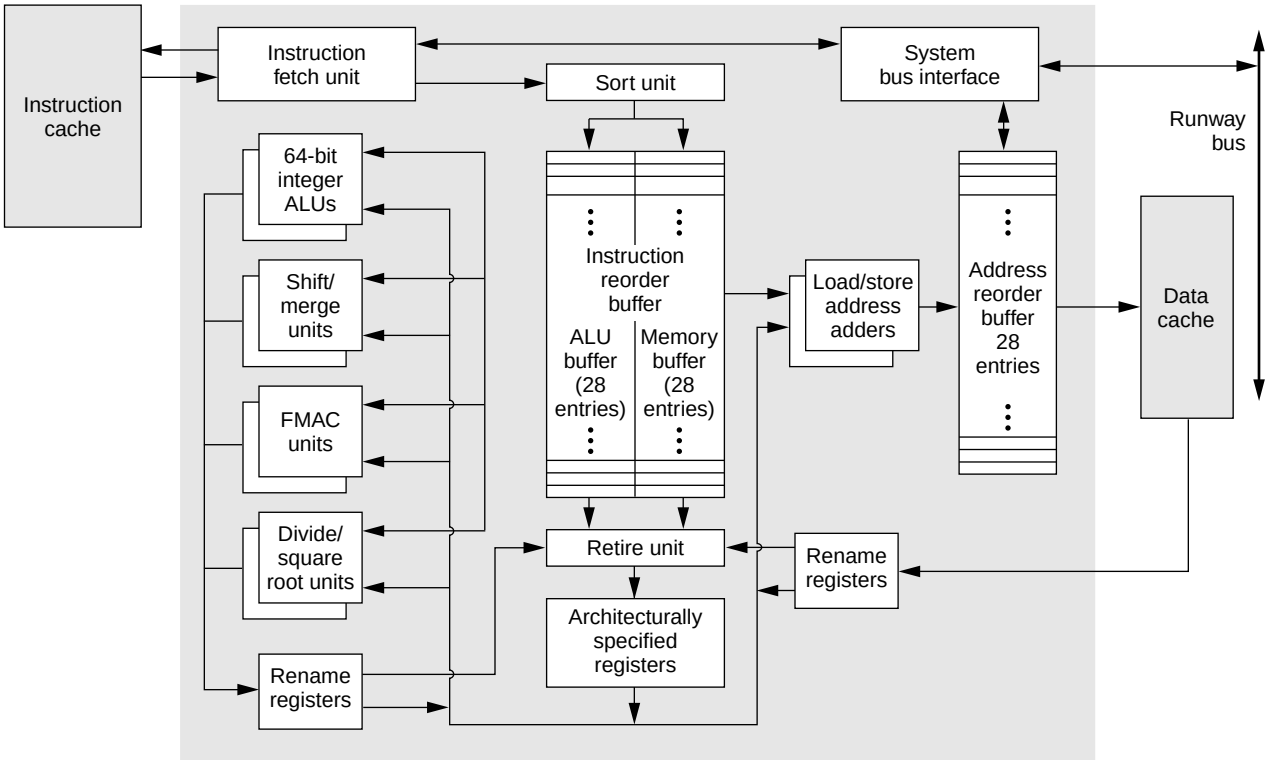

Figure 1. Functional block diagram of the PA-8000.

The PA-8000 executes at a peak rate of four instructions per cycle, enabled by a large complement of computational units, shown at the left side of Figure 1. For integer operation, it includes two 64-bit integer ALUs and two 64-bit shift/merge units. All integer functional units have a single-cycle latency. For floating-point applications, the chip includes dual floating-point multiply-and-accumulate (FMAC) units and dual divide/square root units. We optimized the FMAC units for performing the common operation A times B plus C. By fusing an add to a multiply, each FMAC can execute two floating-point operations in just three cycles. In addition to providing low latency for floating-point operations, the FMAC units are fully pipelined so that the PA-8000's peak throughput is four floating-point operations per cycle. The two divide/square root units are not pipelined, but other floating-point operations can execute on the FMAC units while the divide/square root units are busy. A single-precision divide or square root operation requires 17 cycles; a double-precision operation requires 31 cycles.

Such a large array of computational units would be pointless if they could not obtain enough data to operate on. To that end, the PA-8000 incorporates two complete load/store pipes, including two address adders, a 96-entry dual-ported TLB, and a dual-ported cache. The right side of Figure 1 shows the dual load/store units and the memory system interface. The symmetry of dual functional units throughout the processor allows a number of simplifications in data paths, control logic, and signal routing. In effect, this duality provides separate even and odd machines.

As pipelines get deeper and a processor's parallelism increases, instruction fetch bandwidth and branch prediction become increasingly important. To increase fetch bandwidth and mitigate the effect of pipeline stalls for predicted-taken branches, the PA-8000 incorporates a 32-entry branch target address cache. The BTAC is a fully associative structure that associates a branch instruction's address with its target's address. Whenever the processor encounters a predicted-taken branch in the instruction stream, it creates a BTAC entry for that branch. The next time the fetch unit fetches from the branch's address, the BTAC signals a hit and supplies the branch target's address. The fetch unit can then immediately fetch the branch's target without incurring a penalty, resulting in a zero-state taken-branch penalty for branches that hit in the BTAC. To improve the hit rate, the BTAC holds only predicted-taken branches. If a predicted-untaken branch hits in the BTAC, the entry is deleted.

To reduce the number of mispredicted branches, the PA-8000 implements two modes of branch prediction: dynamic and static. Each TLB entry contains a bit to indicate which mode the branch prediction hardware should use; therefore, the software can select the mode on a page-by-page basis. In dynamic mode, the instruction fetch unit consults a 256-entry branch history table, which stores the results of each branch's last three iterations (either taken or untaken). The instruction fetch unit predicts that a given branch's outcome will be the same as the majority of the last three outcomes. In static prediction mode, the processor predicts that most conditional forward branches will be untaken and that most conditional

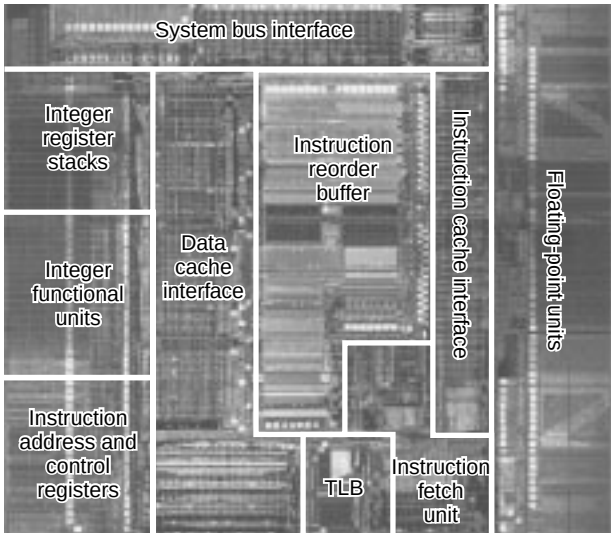

Figure 2. The PA-8000 die.

backward branches will be taken. For the common compare-and-branch instruction, the PA 2.0 architecture defines a branch predict bit that indicates whether the processor should follow this normal prediction convention or the opposite convention. Compilers using either heuristic methods or profile-based optimization can use the static prediction mode to effectively communicate branch probabilities to the hardware.

Cache design

The PA-8000 uses separate, large, single-level, off-chip, direct-mapped caches for instructions and data. It supports up to 4 Mbytes for instructions and 4 Mbytes for data, using industry-standard synchronous SRAMs. We provided two complete copies of the data cache tags so that the processor can accommodate two independent accesses, which need not be to the same cache line.

Why did we design the processor without on-chip caches? The main reason is performance. Competing designs incorporate small on-chip caches to enable higher clock frequencies. Small on-chip caches support benchmark performance but fade in large applications, so we decided to make better use of the die area. The sophisticated instruction reorder buffer allowed us to hide the effects of a pipelined two-state cache latency. In fact, our simulations demonstrated only a 5% performance improvement if the cache was on chip and had a single-cycle latency. A flat cache hierarchy also eliminates the design complexity associated with a two-level cache.

Physical chip characteristics

We fabricated the PA-8000 in HP's 0.5-micron, 3.3-V CMOS process. Although the drawn geometries are not very aggressive, we still obtain a respectable 0.28-micron effective transistor-channel length. In addition, we invested heavily in the design process to ensure that both layout and circuits would scale easily into more advanced technologies with smaller geometries. The chip contains five metal layers: two for tight pitch routing and local interconnections, two for low-RC (resistance-capacitance) global routing, and a final layer for clock and power supply routing.

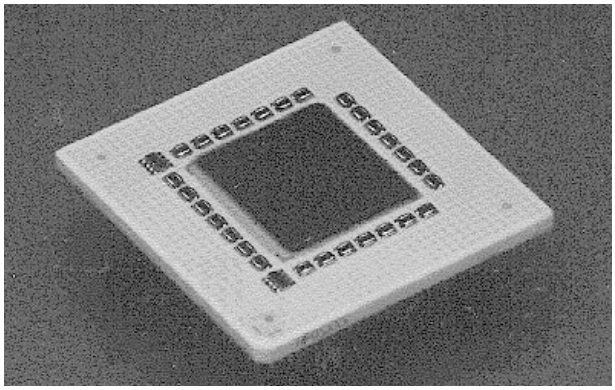

Figure 3. The PA-8000 package.

The processor design includes a three-level clock network, organized as a modified H-tree. The clock syncs serve as primary inputs, received by a central buffer and driven to 12 secondary clock buffers located at strategic spots around the chip. These buffers then drive the clock to the major circuit areas, where it is received by clock "gaters" (third-level clock buffers) featuring high gain and a very short in-to-out delay. The approximately 7,000 gaters can generate many clock "flavors": two-phase overlapping or nonoverlapping, inverting or noninverting, qualified or nonqualified. The clock qualification is useful for synchronous register sets and dumps, as well as for powering down sections of logic not in use. To minimize clock skew and improve edge rates, we simulated and tuned the clock network extensively. The simulated final clock skew for this design was no greater than 170 ps between any points on the die.

Under nominal operating conditions of 20°C room temperature and 3.3-V power supplies, the chip can run at up to 250 MHz. The die measures 17.68 mm × 19.1 mm and contains 3.8 million transistors. Approximately 75% of the chip is either full- or semicustom. Figure 2 shows a photograph of the die with all major areas labeled. The instruction reorder buffer in the center of the chip provides convenient access to all functional units. The left side of the chip contains the integer data path; the right side, the floating-point data path.

Flip-chip packaging technology enables the chip to support a large number of I/Os—704 in all. In addition to the I/O signals, 1,200 power and ground bumps connect to the 1,085-pin package via a land grid array. There are fewer pins than total I/Os and bumps because each power and ground pin can connect to multiple bumps. Figure 3 shows the packaged part. Solder bump interconnections attach the chip to the ceramic carrier, which is mounted on a conventional PC board. This packaging has several advantages. Obviously, the high pin count enables wide off-chip caches. The ability to place I/Os anywhere on the die improves area utilization and reduces on-chip RC delays. Finally, the low inductance of the signal and power supply paths reduces noise and propagation delays.

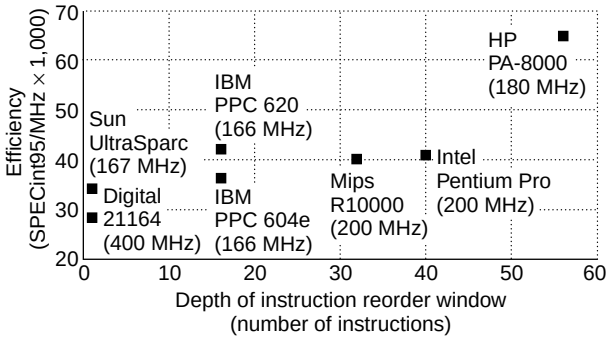

Figure 4. Performance effect of instruction reordering. These examples are four-way, superscalar microprocessors, except the Pentium Pro, which has a three-way design; PPC: Power PC. (Source: *Microprocessor Report*, Apr. 15, 1996)

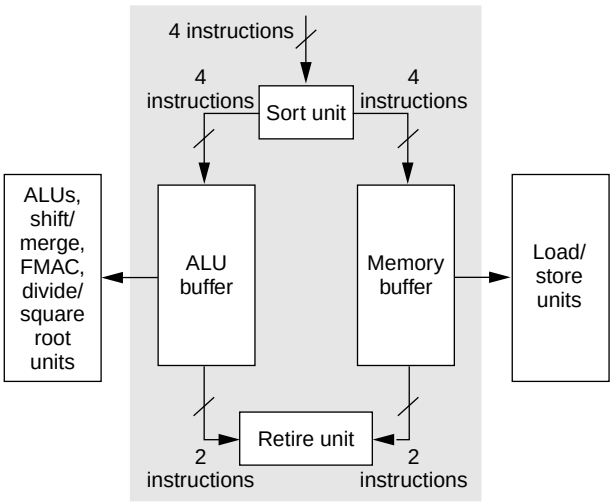

Figure 5. The PA-8000's instruction reorder buffer.

Performance

At 180 MHz with 1-Mbyte instruction and data caches, the PA-8000 achieves benchmark performance of over 11.8 SPECint95 and over 20.2 SPECfp95, and thus was the world's fastest processor when system shipments began in January 1996. A four-way PA-8000 multiprocessor system available in June 1997 has produced 14,739.03 tpmC (the Transaction Processing Performance Council's benchmark C) at a price of $132.25 per tpmC.

Several distinguishing features enable the PA-8000 to achieve this level of performance:

- First, the processor includes a large number of functional units—10. However, sustaining greater than two-way superscalar operation demands advanced instruction-scheduling methods to supply a steady stream of independent tasks to the functional units. To achieve this, we incorporated an aggressive out-of-order execution capability. The instruction reorder buffer provides a large window of available instructions combined with a robust dependency-tracking system.
- Next, explicit compiler options that generate hints to the processor help performance a great deal. The processor uses these special instructions to prefetch data and to communicate statically predicted branch behavior to the branch history table.
- Finally, the system bus interface can track up to 10 pending data cache misses plus an instruction cache prefetch. Servicing multiple misses in parallel reduces the average performance penalty caused by each miss.

Instruction reorder buffer

Because of restrictions on compiler scheduling, a key decision in our design was to have the PA-8000 perform its own instruction scheduling. It was for this task that we equipped it with the 56-entry instruction reorder buffer. The IRB consists of the ALU buffer, which can store up to 28 computation instructions, and the memory buffer, which can hold up to 28 load and store instructions. These buffers track the interdependencies of the instructions they contain and allow instructions anywhere in the window to execute as soon as they are ready.

As a special feature, the IRB tracks branch prediction outcomes, and when it identifies a misprediction, flash-invalidates all incorrectly fetched instructions. Fetching then resumes down the correct path without further wasted cycles.

The IRB serves as the central control logic for the entire chip, yet it consists of only 850,000 transistors and consumes less than 20% of the die area. Because today's compilers lack the visibility of runtime information useful for optimal scheduling, the IRB is of paramount importance to microprocessor performance. The graph in Figure 4 correlates microprocessor efficiency, based on SPECint per MHz, with instruction reorder window depth. As should come as no surprise, the larger the buffer, the better the performance. The PA-8000's IRB is 40% larger than that of the nearest competitor.

Instruction reordering also breaks through another bottleneck: memory latency. Although the dual load/store pipes keep the computation units busy as long as the data is cache-resident, a data cache miss might still cause a disruption. But with instruction reordering, execution can continue for many cycles on instructions that do not depend on the data cache miss. Because the IRB can hold so many instructions, the PA-8000 can execute instructions well past the missed load or store. This ability to accomplish useful work during a data cache miss significantly reduces its impact on performance.

The large window of available instructions also allows the overlap of multiple data cache misses. If the processor detects a second data cache miss while an earlier miss is still being serviced by main memory, it issues the second miss to the system bus as well.

The life of an instruction

Figure 5 diagrams the PA-8000's IRB. Instructions enter through the sort block, which, on the basis of instruction type, routes them to the appropriate IRB section, where they

remain until they retire. Functional units are connected to the IRB sections appropriate to the types of instructions they execute. After instructions execute, the retire block removes them from the system.

Instruction insertion. To maximize the IRB's chances of finding four instructions that are all ready to execute on a given cycle, it always must be as full as possible. Therefore, we built a high-performance fetch unit for the PA-8000. This unit fetches, in program order, up to four instructions per cycle from the single-level, off-chip instruction cache.

The fetch unit performs limited predecoding and inserts the instructions round-robin into the appropriate IRB section. Each IRB section must be able to handle four incoming instructions per cycle, since there are no restrictions on the mix of instructions inserted.

In two special cases, a single instruction is not associated with a single IRB entry: 1) Although the processor executes branches from the ALU buffer, it also stores them in the memory buffer as placeholders to indicate which entries to invalidate after a mispredicted branch. 2) Instructions with both a computation and a memory component and two targets, such as the load-word-and-modify (LDWM) instruction, split into two pieces and occupy an entry in both IRB sections.

Instruction launch. The IRB allows instructions to execute out of order. During every cycle, both IRB buffers allow the oldest even and the oldest odd instruction for which all operands are available to execute on the functional units. Thus, up to four instructions can execute at once: two computation and two memory reference instructions. Once an instruction has executed, a temporary rename register holds its result and makes it available to subsequent instructions.

Instruction retire. Instructions retire from the IRB in program order once they have executed and any exceptions have been detected. This gives software a precise exception model. As instructions retire, the contents of the rename registers transfer to the general registers, stores enter a queue to be written to cache, and instruction results commit to the architecturally specified state. The retire unit can handle up to two ALU or floating-point instructions and up to two memory instructions each cycle.

Dependency tracking

The IRB solves many scheduling problems usually encountered in a superscalar machine. It matches instructions to available functional units by scanning a sufficiently large number of instructions at once to find those that are ready to execute. Tracking dependencies among these instructions and determining which may execute are complications inherent to an out-of-order machine. The PA-8000 manages over a dozen types of dependencies for as many as 56 instructions, identifying all possible instruction dependencies at insertion.

Operand dependencies. Operand dependencies occur when an instruction's source data is the result of an earlier instruction. Because of their high frequency, the IRB tracks operand dependencies using a high-performance broadcast mechanism for maximum performance. Upon insertion, a two-pass mechanism determines the most recent writers for each incoming instruction's source operands. Each instruction then records whether it should obtain a given operand from a specific rename register or from the general registers (no dependency). When an instruction receives permission to execute (launch), it broadcasts its rename register number to all other IRB entries. If a later instruction has a source operand tag that matches one driven on the launch bus, it can clear that dependency. All of this takes place in a single cycle, enabling a dependent instruction to execute in the very next cycle after a producer instruction executes.

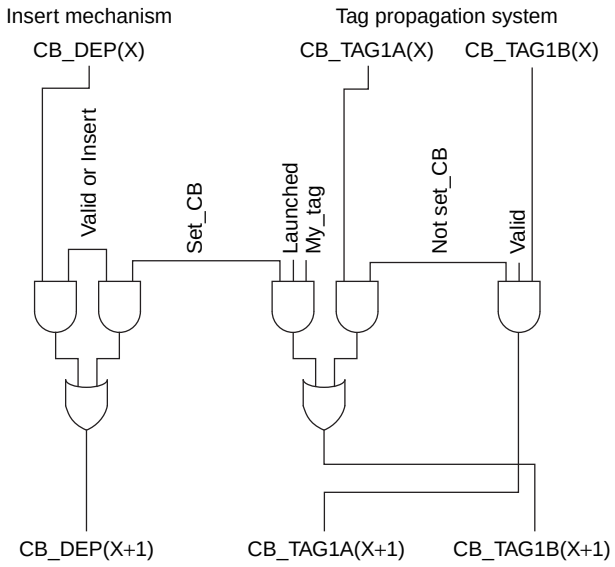

Figure 6. Bit slice for carry borrow dependencies.

The IRB sends the functional units appropriate information about the launching instructions. This ensures that source operands are routed from the correct location (bypassed from the functional units or delivered by the rename or general registers) and that results are stored in the appropriate rename register. By the time an instruction launches, the result of a producing instruction may have moved from the rename registers to the general registers. The IRB, however, does not track this information; the execution units detect it on the fly through a color bit comparison of the rename register. (A color bit is a special bit for detecting a match between valid and invalid data.) Every time an instruction retires, the color bit of the associated rename register flips.

Carry borrow dependencies. The PA-8000 uses a different tracking method for CB dependencies, embodied in the circuitry shown in Figure 6. An instruction that uses the CB bits of the processor status word has a CB dependency. Most arithmetic operations set these bits. Like operand dependencies, all CB dependencies are identified at insertion time. The difference is that although an instruction knows it has a dependency, it does not know which instruction it depends on. Rather, it receives information that some previous instruction affects the portion of the processor status word that concerns it. The last valid IRB entry maintains the information, for use by incoming instructions, that the IRB still contains a valid CB-setting instruction. During instruction insertion, this information passes through up to four con-

secutive IRB entries, the maximum number of instructions that can enter during one cycle. Although this method is conceptually simple, it requires complex control logic for many anomalous cases—for example, the last valid IRB entry's retiring just as new instructions enter.

Once an instruction that sets the CB bits receives permission to execute, the launching IRB entry sends a signal to the entries below it that its data is now available. The launching instruction's slot number also passes along on buses that indicate the most recent writer of the CB bits. While an instruction that sets the CB bits (most arithmetic instructions) is waiting to execute, it blocks these buses. An instruction that does not change these bits simply passes on the information that it received from the entry above it. This information usually propagates at the rate of two IRB entries per cycle. Once a dependent instruction receives an indication that the last instruction that writes the bits it uses has executed, it can clear its dependency. When it gets permission to launch, it drives out the information it has stored about the most recent writer of the CB bits. Then, either a rename register or the processor status word dumps the correct information to the execution unit.

In contrast to the full broadcast mechanism for clearing operand dependencies, this propagation system sacrifices performance to save area in certain cases. CB dependent instructions occur comparatively infrequently, and the dependency-clearing mechanism requires only about one-third the area of the broadcast method. Most important, no performance implication results from the common case in which an instruction that uses CB information follows immediately an instruction that sets it.

Address dependencies. The address reorder buffer schedules accesses to the dual-ported data cache much as the IRB schedules instructions for the functional units. After one of the address adders calculates an address, it enters a corresponding slot in the ARB, which contains one slot for each slot in the memory buffer. The ARB then launches the oldest even double-word address and the oldest odd double-word address to the data cache each cycle. The ARB unit also contains comparators that detect store-to-load dependencies. They also detect that a load or store has missed a cache line for which a request has already been sent to main memory, thus avoiding a second request.

THE PA-8000 RISC CPU incorporates aggressive out-of-order execution, intelligent performance and area trade-offs, and balanced hardware utilization. Dual load/store pipes ensure an adequate supply of data to the multiple computation units. Instruction reordering ensures that data dependencies do not become a bottleneck. The high-performance system bus supports multiple cache misses, thus reducing the effects of data cache miss penalties. The branch target address cache and branch history table counteract branch penalties. These features add up to an innovative microprocessor suitable for both technical and commercial applications. The chip is currently in production, and systems have been shipping since January 1996.

Hewlett-Packard is developing two follow-on chips, the PA-8200 and the PA-8500. The PA-8200 will have a higher frequency, a 120-entry TLB, a 1,024-entry branch prediction cache, and instruction and data caches of 2 Mbytes each. We expect technical and commercial applications to experience a 35% to 75% performance improvement, and projected ratings are 15.5 SPECint95 and 25 SPECfp95. We are also developing a number of memory system enhancements. Systems should be shipping by June 1997. Plans for the PA-8500 processor include an improved microarchitecture, a much higher clock speed, and a 0.25-micron process.

Acknowledgments

The author acknowledges the contributions of the technical and management staff at Hewlett-Packard laboratories in Fort Collins, Colorado, and Cupertino and Roseville, California. These include the Engineering Systems, General Systems, Enterprise Systems, Operating Systems, Compiler, and System Performance laboratories. Without the tremendous effort of so many individuals, the PA-8000 would not have been possible.

Suggested readings

Hunt, D., "Advanced Performance Features of the 64-Bit PA-8000," *Proc. Compcon*, IEEE CS Press, Los Alamitos, Calif., 1995, pp. 123-128.

Lotz, J., et al., "A Quad-Issue Out-of-Order RISC CPU," *Proc. Int'l Solid-State Circuits Conf.*, IEEE, Piscataway, N.J., 1996, pp. 210-211.

Gaddis, N., et al., "A 56-Entry Instruction Reorder Buffer," *Proc. Int'l Solid-State Circuits Conf.*, IEEE, Piscataway, N.J., 1996, pp. 212-213.

Ashok Kumar is a hardware design engineer at Hewlett-Packard's Systems Performance Lab in Cupertino, California, where he develops performance simulators and helps design future processor architectures. Earlier, he was a block designer for the PA-8000 instruction reorder buffer at HP's Engineering Systems Lab in Fort Collins, Colorado.

Kumar received his MSEE from Stanford University and his BSEE from the University of Washington. He is a member of the IEEE, the Computer Society, Tau Beta Pi, and Eta Kappa Nu.

Send correspondence about this article to the author at Hewlett-Packard Co., 11000 Wolfe Rd., MS 42LX, Cupertino, CA; ashok@cup.hp.com.