PA-RISC 8x00 Family of Microprocessors with Focus on PA-8700

Technical White Paper

April 2000

Table of Contents

List of Figures

List of Tables

1. PA-RISC: Betting the House and Winning

In 1983 Hewlett-Packard's computing business decided to embark on a risky proposition: unite the different processors and operating systems the company used in its three computer lines (the HP1000, HP3000, and HP9000) into a single, highly scalable, hardware platform. The decision to take this challenge on was a big bet for Hewlett-Packard. In essence Hewlett-Packard was counting on successfully designing a new, RISC (Reduced Instruction Set Computing) processor architecture that could serve the various and divergent interests within. Hewlett-Packard was also committing to investing in its own flavor of Unix that would take advantage of the new RISC processor and provide a mission-critical platform for its enterprise customers.

The results of this engineering challenge have been extraordinary. Over the years Hewlett-Packard has developed a highly successful RISC processor line – the PA-RISC (Precision Architecture Reduced Instruction Set Computing) family, in addition to a highly robust and functional, Unix operating system - HP-UX. PA-RISC and HP-UX have been inseparable in Hewlett-Packard's award-winning, Unix workstations and high-end Unix servers (N-class, V-class, L-class, A-class). Hewlett-Packard's commitment and execution have been stellar. For example, beginning with the PA-8000 through the PA-8600, each new PA-RISC processor has set the standards for excellence, in terms of performance and scalability, since the time of introduction. Hewlett-Packard has been at the forefront of the RISC processor space for over a decade, and is continuing to work to maintain its lead.

PA-RISC processors can be used in a myriad of configurations from uni-processor to 64 way multi-processors. Hewlett-Packard's enterprise solutions allow customers to configure their systems in SMP (Symmetric Multi-Processor) or ccNUMA (Non-Uniform Memory Access) formats, providing scalability and flexibility to customers.

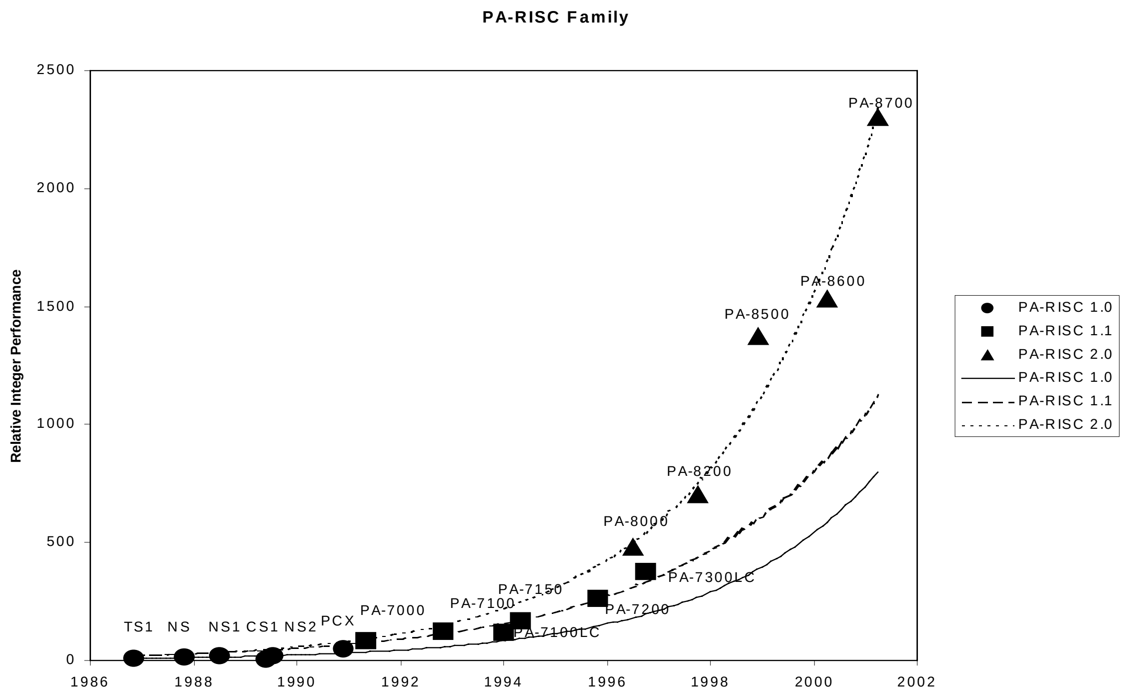
This white paper will describe Hewlett-Packard's PA-RISC design philosophy and PA-RISC family evolution. It will highlight the latest PA-RISC addition, the PA-8700, expected to ship in HP systems in 2001. The PA-8700 continues the legacy of the PA-RISC family by delivering on new features and enhancements over its predecessor, the PA-8600, and is the first PA processor to use .18 µm, silicon-on-insulator, copper process technology. This paper will conclude by demonstrating the continued viability of the PA-RISC family in light of the much talked-about IA-64 processor architecture – a co-invention with Intel, and how Hewlett-Packard will smoothly transition its customers to IA-64.

2. RISC Philosophy

Hewlett-Packard introduced its first PA-RISC computer in 1985. It was built using off-the-shelf TTL (transistor-to-transistor logic) parts and had an 8 MHz clock. In the ensuing years Hewlett-Packard has designed 16 different PA-RISC processor implementations. These PA-RISC processors have been implemented in TTL, NMOS, and 7 different CMOS processes. No other RISC vendor has designed as many RISC processors or had their designs implemented in as many different process technologies as the PA-RISC family. PA-RISC continues to be a staple of Hewlett-Packard's enterprise systems.

Over this period of time Hewlett-Packard has constantly worked to design processors delivering outstanding performance, reliability, and cost effectiveness. Figure 1 shows the progress that has been made in system performance. In order to achieve these results, Hewlett-Packard has made investments in 3 main areas: clock frequency, VLSI (Very Large Scale Integration) integration, and architectural enhancements.

Figure 1 PA-RISC Family Evolution



Clock Frequency

The frequencies, at which PA-RISC processors operate, are a prime determinant of performance. These frequencies have increased by nearly 2 orders of magnitude over the last 15 years; from 8 MHz in the mid-80s to 550 MHz for Hewlett-Packard's newest currently shipping part—the PA-8600. Hewlett-Packard will continue to increase clock frequencies of the PA-RISC family to 1 GHz and beyond.

VLSI Integration

A high level of integration provides many benefits. Higher levels of integration provide better reliability, because the number of package connections on a board (these tend to be frequent failure points) are minimized. Higher levels of integration also provide higher performance, because it avoids the large speed penalty required to drive a signal off one chip, down a printed circuit board trace, and onto another chip. Furthermore, because the number of pins available on a package is limited, it is often necessary to limit the width of off-chip interfaces. Power consumption is also reduced, because chip crossings tend to be slow and consume large amounts of power. Finally, higher levels of integration promote lower cost due to lower part counts and savings in board area and power.

The first PA-RISC product required 6 boards to implement using small and medium-scale integration parts. The first VLSI implementation required 8 custom VLSI parts. Today, the entire processor, including large amounts of on-chip cache memory, is integrated into a single VLSI chip. This level of integration is possible because we can now pack more than 3 orders of magnitude more transistors on a single die compared to the first PA-RISC VLSI implementations.

Architectural Enhancements

There are two levels of architectural enhancements: system-level and implementation-level. System-level architecture concerns those features of a machine visible to an assembly level programmer, involving compilers and binary compatibility. Examples of changing the system-level architecture would be adding an instruction or increasing the number of programmer visible registers. Implementation-architecture concerns the internal organization of the processor and is generally invisible to a programmer with the exception of how quickly a program executes. Examples of changing the implementation-level architecture would be increasing the cache size or adding superscalar capability.

Because system level architecture changes are visible to a programmer, any such changes have the potential to adversely affect a customer. For this reason, Hewlett-Packard rarely changes the system-level architecture of its PA-RISC processors. When Hewlett-Packard does make a system level architecture change, it goes to great efforts to ensure that customers will not be negatively affected by the changes. On the other hand, implementation architecture is invisible to the user. For each new design, Hewlett-Packard aggressively investigates what implementation-level architecture changes can be made to maximize the performance and minimize the cost.

Hewlett-Packard has made two changes to the implementation-level architecture of the PA-RISC family. The initial PA-RISC designs were done following the PA-RISC 1.0 specification. Beginning with the PA-7000 in 1991, designs were done following the PA-RISC 1.1 specification. The most important of the changes introduced in PA-RISC 1.1 were:

- Increasing the page size from 2 Kbytes to 4 Kbytes
- Adding 16 more floating point registers
- Increasing the speed and efficiency of trap handlers.

All changes were made in a backward-compatible manner so any user-level code written to the PA-RISC 1.0 specification would continue to run unchanged on a PA-RISC 1.1 machine. These system-level architectural changes allowed more efficient use of large memories, faster floating point execution, and higher performance of any code which depended on traps including many operating system functions. By carefully making these changes to ensure backward compatibility, the customer received the benefit of higher performance without being forced to make code changes.

In 1996, Hewlett-Packard introduced the PA-8000 family, which utilized the PA-RISC 2.0 specification. The most important changes introduced in PA-RISC 2.0 were:

- Including 64-bit extensions for both data and addresses
- Offering multimedia extensions
- Using branch prediction via “hints”
- Implementing weak memory ordering
- Supporting coherent I/O
- Including an FMAC (Multiply-accumulate)

- Utilizing cross space branches/shared library acceleration.

The PA-RISC 2.0 architecture provides full support for 64-bit computing with 64-bit registers and datapath and full support for 64-bit integers as well as providing a 32-bit mode for backward compatibility. The architecture also provides a flat 64-bit virtual address space and can support physical addresses greater than 32 bits. These changes enabled customers to work with much larger code and data sets, and to realize higher performance. As with the transition to PA-RISC1.1, Hewlett-Packard went to great efforts to make the changes in a backward-compatible manner to protect customers investments in software.

3. Evolution of the PA-RISC Family

Over the last 15 years a large number of new implementation level architecture techniques have been coupled with the two system level architecture changes to optimize performance and minimize cost. Reviewing Figure 1, dramatic increases in performance are evident over this period of time. The PA-RISC 1.0 implementations all utilized relatively simple pipelines. Performance improvements largely came by increasing the clock frequency and improving cache performance. In the early 90s, higher levels of integration made it possible to build a significantly more sophisticated processor and thus improve performance. In the PA-RISC 1.1 chips (PA-7xx0 family) frequency continued to increase with each new iteration just as it had for the PA-RISC 1.0 chips. But to achieve the still higher levels of performance required by customers, additional techniques were employed to improve performance beyond what frequency increases alone could provide. Superscalar execution (executing more than one instruction at a time) was one such technique used to improve performance. This trend began in 1992 with the PA-7100, which had a limited ability to launch 2 instructions at once. Because none of the functional units were duplicated, only an integer instruction and a floating-point instruction could execute together. There was not enough parallel hardware to launch two integer instructions or two floating point instructions at once, and therefore, the performance gain was modest. The PA-7100LC in 1993 and the PA-7200 in 1994 both had a second integer unit, which gave them the flexibility to execute two integer instructions together. This ability allowed these processors to achieve as much as a 20% increase in throughput on real world applications relative to a PA-7100 processor operating at the same frequency.

By the mid 90s, continued increases in the level of integration provided still more opportunities for higher performance. It was now possible to place large numbers of functional units on one chip. If these functional units could all be kept busy, significant performance gains would be possible. Hewlett-Packard conducted extensive performance studies looking at the impact of adding large numbers of functional units to a processor. In theory, large gains were possible. But by studying important customer applications, it was determined that in practice, gains would be minimal. In the real world, data and control dependencies in typical programs severely limited the additional performance gained by just adding more functional units.

In order to further increase the number of instructions that can be executed together, a completely new micro-architecture was required. The PA-8000, introduced in 1996, represented such a design, providing a significant boost in performance relative to its predecessors. A PA-8000 processor was able to deliver a 30% or more increase in throughput on real world applications relative to a PA-7200 processor operating at the same frequency. Some of the advanced micro-architectural features of the PA-8x00 family are described below.

The trend lines for PA-RISC 1.0, PA-RISC 1.1, and PA-RISC 2.0 plotted on Figure 1 show the progress that was made as designs moved from one generation to the next. The trend line for PA-RISC 1.0 is flatter and lower then the trend line for PA-RISC 1.1. Likewise, the trend line for PA-RISC 1.1 is flatter and lower than the trend line for PA-RISC 2.0. Improvements were made with each generation to move to a more aggressive performance curve.

Looking to the future, Hewlett-Packard will continue to offer higher performance members of the PA-8x00 family.

4. Key Highlights from PA-8000 to PA-8600

The design goals of the PA-8x00 family are to:

- Lead the industry in application performance
- Provide full binary compatibility with all existing PA-RISC binaries
- Extend the robust PA-RISC design to even higher reliability levels
- Deliver maximum performance on binaries already tuned for previous members of the PA-8x00 family.

The PA-8000 introduced in 1996 was the first member of the PA-8x00 family. The PA-8000 was a totally new PA-RISC design and broke new ground in the following areas:

- PA-RISC 2.0 specification, including 64-bit computing
- Large complement of functional units
- Support for large dual-ported data caches
- Extensive branch prediction hardware
- Sophisticated instruction re-ordering hardware.

Together, these features allowed the 64-bit PA-8000 to effectively exploit instruction level parallelism and set new standards for performance. The PA-8000 supported 40 bits of physical address—enough to address 1 Terabyte of physical memory enabling customers to work with very large data and code sets.

The PA-8200 introduced in 1997 improved on the PA-8000 by:

- Increasing frequency to 240 MHz
- Quadrupling the size of the Branch History Table (BHT) from 256 entries to 1024
- Improving the branch prediction algorithms
- Increasing the number of Translation Look-aside Buffer (TLB) entries from 96 to 120
- Doubling the size of off-chip caches supported from 1 Mbyte to 2 Mbytes.

These improvements allowed the PA-8200 to continue to set the standard for performance.

The next processor in the PA-8000 family was the PA-8500 introduced in 1998. The PA-8500 delivered significantly higher levels of performance by:

- Increasing clock frequency to 440 MHz
- Integrating large 4-way set-associative primary caches on the chip (1MB data cache and .5MB instruction cache)
- Doubling the number of BHT entries from 1024 to 2048
- Increasing the number of TLB entries from 120 to 160
- Increased the system-bus bandwidth.

High frequency operation and large primary caches were both achieved by implementing the PA-8500 in a 0.25 µm CMOS process. The PA-8500 offered unprecedented levels of performance, while reducing system costs, a rare combination.
In 2000, Hewlett-Packard introduced the newest member of the PA-8000 family—The PA-8600. The new PA-8600 features include:

- Increased clock frequency to 550 MHz
- A quasi LRU replacement policy for the instruction cache

- Optimization of the order that certain bus transactions occur.

The PA-8600 provides industry leadership performance at introduction, just as its predecessors in the PA-8x00 family did. Together, these features will continue to provide industry leadership performance for the PA-RISC family. The next section will delve into the details of the PA-8700 microprocessor.

5. PA-8700 Feature Set Highlights

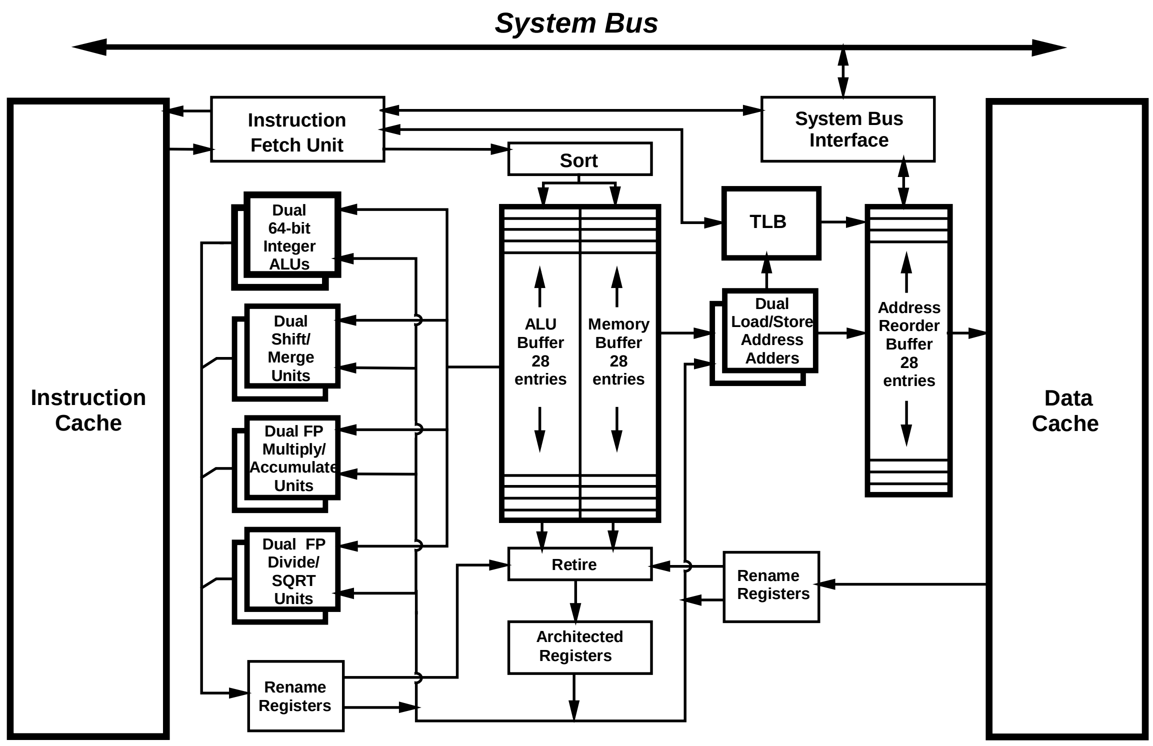
The PA-8700 is the next processor in the PA-8x00 line and leverages the industry award winning PA-8500 design (Microprocessor Report, 1998). Hewlett-Packard will begin shipping products using the PA-8700 in 2001. Notable features provided by the PA-8700 include:

- Clock frequency greater than 800 MHz
- 50% increase in the size of the data cache to 1.5 Mbytes
- 50% increase in the size of the instruction cache to .75 Mbytes
- Data pre-fetching capability
- Quasi LRU replacement policy for the data cache—in addition to the instruction cache quasi LRU carried over from the PA-8600
- Support for 44-bit physical addresses—enough to address 16 Terabytes.

With its introduction in systems in 2001, the PA-8700 will once again continue the industry-leading performance in Hewlett-Packard’s enterprise servers and technical computing products.

The advanced micro-architecture of the PA-8700 aggressively executes as many instructions as possible each cycle to maximize performance. The processor exploits techniques such as out of order execution, speculative execution, and non-blocking caches. Figure 2 is a block diagram of the PA-8700 showing the major functional blocks. The Instruction Fetch Unit can fetch 4 instructions each cycle from the instruction cache. From there, the instructions are forwarded to the Sort unit which places instructions into either the 28 entry ALU Reorder Buffer or the 28 entry Memory Reorder Buffer depending on the instruction type. The Reorder Buffers constantly scan their contents looking for instructions that are ready to execute. The Reorder Buffers track all dependencies between instructions and allows data flow execution across all instructions in the buffers. Instructions are no longer constrained to execute in program order. Instead, as soon as an instruction is identified as ready to be executed, and an appropriate functional unit is available, the instruction is dispatched to the functional unit to begin execution. Each cycle, up to four instructions can be dispatched. The PA-8700 includes 10 functional units to maximize the number of instructions that can be executed each cycle. These functional units consist of 4 integer units (2 Arithmetic/Logic Units and 2 Shift Merge Units), 4 floating point units (2 Multiply and Accumulate Units and 2 Divide/Square Root Units), and 2 Load/Store Units (one for even double-word addresses and one for odd double-word addresses). After instructions have completed execution in the various functional units, they return to the Retire Unit to update the architected state of the processor and thus complete execution.

Figure 2 PA-8700 Block Diagram



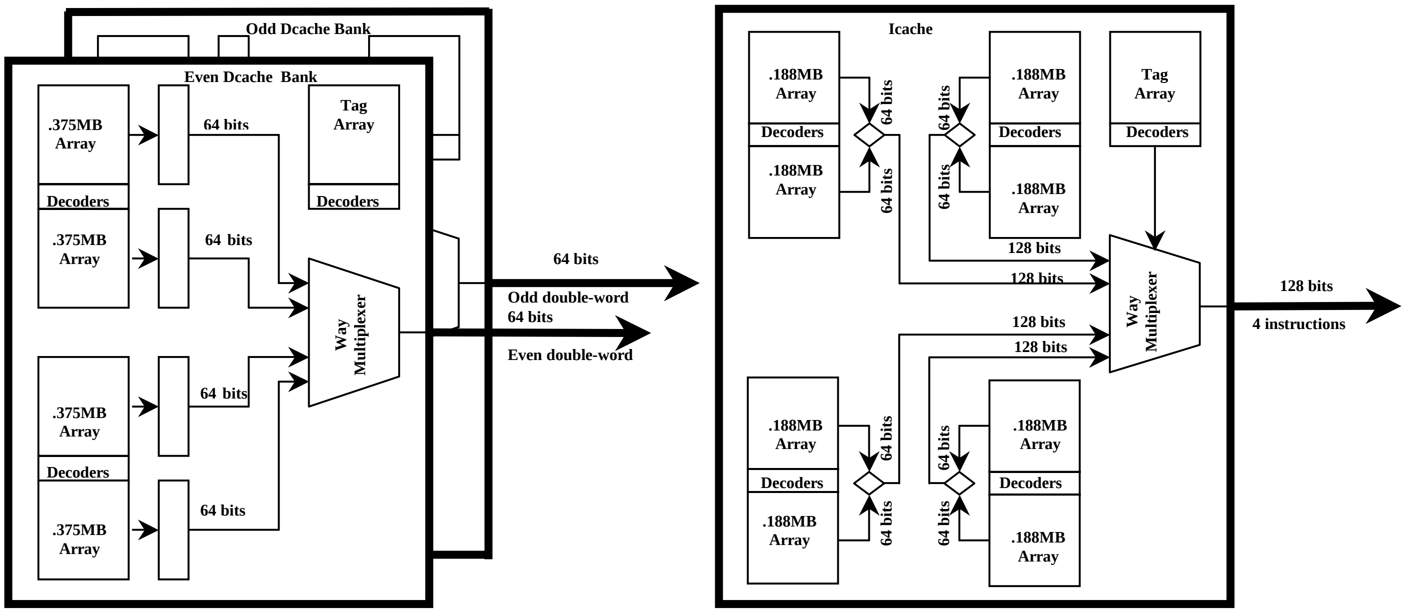
Large Robust Primary Caches

Members of the PA-8x00 family have always used large high performance caches. The advantages of large caches are obvious. Small on-chip level one (L1) caches backed by larger and much slower off-chip level two (L2) cache may perform adequately for small benchmarks. But when faced with more demanding applications such as data warehousing, on-line transaction processing, or technical computing, such caches usually result in disappointing performance. The PA-8000 and PA-8200 both utilized off-chip caches because the processes they were implemented in did not support the level of integration necessary for adequate on-chip caches. Starting with the PA-8500, integration levels reached the point where large on-chip L1 caches could be implemented. The PA-8500 set an industry first by including 1.5 Mbytes of on-chip cache. The move to on-chip cache provided several benefits. The expensive and power-hungry external RAM chips used by the PA-8000 and PA-8200 (at least 12 parts) were eliminated, slashing costs and dramatically reducing power. The move to on-chip cache also reduced the PA-8500's I/O count from almost 1100 signals to 550 signals. This improved system reliability while making it possible to run at much higher frequency than would be possible with off-chip caches.

The PA-8000 and PA-8200 utilized direct-mapped caches—meaning that for any given address, there is only one location where the data can possibly reside. In contrast, an *n*-way set-associative cache has *n* possible locations where an address' data can reside. An *n*-way set-associative cache performs better than a direct-mapped cache of the same size. The larger the value of *n* in an *n*-way set-associative cache, the better the performance. Unfortunately, an *n*-way set-associative cache also requires *n* times as many signals to the cache compared with a direct-mapped cache. As a result, implementing an off-chip set-associative cache would have been cost prohibitive for the PA-8000/PA-8200 due to the number of I/Os required. By moving the cache on chip, a set-associative cache becomes practical, as the large number of signals to the cache are now wires on a chip instead of I/Os on a package. The PA-8500 as well as the PA-8600 and PA-8700 use 4-way set-associative caches for both the instruction and data caches. The PA-8700 increases the amount of cache by 50% over the PA-8600 to provide a total of 2.25 Mbytes of on-chip L1 cache. As a result, cache hit rates for even demanding applications are significantly improved on the PA-8700 leading to even higher levels of performance.

The PA-8700 cache organization is shown in Figure 3. The data cache is a four-way set-associative pipelined cache and is constructed of two independent banks. One bank holds data for even double-word addresses and one bank holds data for odd double-word addresses. Each 0.75MB data cache bank is implemented as four arrays which independently provide a double-word of data plus error correction bits. Each array provides one way of associativity. Data is organized within the arrays so a full cache line can be addressed at a time, or four ways of associativity can be addressed together. The cache line tags are held in a smaller and separately addressable RAM array. In this way the data and tag can either be accessed together for a data read, or independently for a data store at the same time another store is accessing its cache line status. The PA-8700 instruction cache is a 0.75MB four-way set-associative pipelined cache that can provide 4 instructions per cycle to the instruction fetch unit. The instruction cache is implemented as four arrays which each provides 128 bits of instruction plus pre-decode bits and parity bits. Each array provides one way of associativity.

Figure 3 PA-8700 Cache Block Diagram



Data Integrity - Error Detection and Error Correction Capabilities

The PA-RISC family has always incorporated error checking and error correction features in their caches to protect customer data. For obvious reasons, data integrity is absolutely critical in creating highly-available systems, and Hewlett-Packard's commitment begins with the processor. All data stored in PA-8700 caches are protected from single-bit errors.

For the instruction cache, simple single-bit parity checking is sufficient, because its contents are always clean (data in cache that hasn't been modified is referred to as 'clean'; data which has been modified by a store instruction is referred to as 'dirty'). Any time an instruction access encounters an error, the access is treated as a cache miss. Cache lines with corrupted data are invalidated, and the correct data is re-accessed from memory.

It takes more effort to protect the data cache, because error correction is required if a dirty line in cache becomes corrupted. The PA-8700 provides six extra bits per word to enable single-bit error correction to protect the cache data. The correction, however, is not directly in the cache access path, where it would lengthen the critical cache access latency. Errors are instead detected by parallel error checking logic. If an error is detected, the corrupted data is forced out of the cache. The data is corrected in the copy out path if the line is dirty. If the line is clean, it is invalidated. The access is then re-executed, causing the line to be brought back into cache with the corrected data.

The data cache's tag contains the physical address of the line and its status. This information is needed to know whether a line needs to be copied back to memory and where in memory it belongs. Therefore, the data cache's tag RAMs are also correctable. The PA-8700 maintains two copies of each cache line's tag to allow two accesses to be serviced simultaneously. Each is parity-protected. If an error is detected in either tag, a copy-out of the line is started. During the copy-out, the tag array that does not have the parity error provides the physical address for memory and the correct status. If the status indicates the line was dirty, the line is copied back to memory. If the line was clean, the cache line is invalidated. Once the cache has been scrubbed in this fashion, the access is executed again to bring the error-free line back into cache.

Static and Dynamic Branch Prediction - Faster Program Execution

Today's high-frequency, superscalar processors pay a high cost every time they incorrectly predict program execution paths. This penalty is high due to increased pipeline depth. But just as importantly, mispredicted branches in program execution prevent the processor from effectively exploiting instruction-level parallelism across branches. There are many approaches to the problem of reducing branch penalties. These can be categorized into static and dynamic methods. Each has its strengths and weaknesses.

- **Static methods**—Static methods rely on the compiler to optimize branch performance. Because the optimizations are performed when the program is compiled, the optimizations do not change as the program runs. These optimizations may be based on execution profile information gathered during a training run (Profile Based Optimization or PBO) or they may be based on the structure of the source code by using heuristics. These optimizations include eliminating certain branches altogether using predication, and encoding hints in branch instructions of the likely branch direction.

  For branches, which are either almost always taken or not taken, compiler hints are very accurate and produce excellent results. The principal advantages of static compiler branch optimizations is that they do not consume chip area and do not have the capacity limits that dynamic branch prediction encounters.

- **Dynamic methods**—In contrast, dynamic branch prediction predicts the future behavior of a branch based on the past behavior of that branch and/or other branches during program execution. Dynamic branch prediction performs better under many circumstances, such as predicting branches based on runtime options and when branches change over time. The main weakness of dynamic branch prediction, is a limited resource for collecting branch statistics.

Because both static and dynamic branch prediction methods have their advantages under different circumstances, the design of the PA-8x00 accommodates both techniques. At compile time, the application developer can mark a binary to be run in one mode or the other depending on which performs better for that program.

The PA-8700's Branch History Table (BHT) is an array of two-bit saturating counters. The counters record whether the branch went in the direction indicated by the compiler's static hint. If the static hint was wrong, the counter is incremented; if correct, the counter is decremented. Each time a branch is fetched, the BHT is consulted and if the counter is zero or one, the static hint encoded in the instruction is followed. If the counter is two or three, the hardware predicts that the branch will go in the opposite direction. These enhancements enable the PA-8700 to combine the advantages of static and dynamic branch prediction methods and are backward compatible with all previous members of the PA-8x00 family.

The PA-8700 has a 56-entry instruction reorder buffer. Instructions are fetched 4 at a time and each instruction is placed in either the ALU buffer or the memory buffer depending on the type of instruction it is. The instruction reorder buffer constantly scans the instructions it contains and as it finds instructions that can execute, they are dispatched to one of the 8 execution units or one of the two load/store pipes. Instructions are no longer constrained to execute in program order. The PA-8700 is also capable of speculative execution. When branch instructions are encountered, the PA-8700 uses sophisticated branch prediction algorithms involving its branch hardware table to guess whether or not the branch will be taken. The machine then speculatively executes instructions from the predicted branch. If the branch prediction was correct the speculatively executed instructions are allowed to retire normally. If the branch was incorrectly predicted all speculatively executed instructions are purged and the processor begins fetching from the correct location.

Profile-Based Optimization – Increase Application Run-time Performance

The radical design of the PA-8000 opened up new opportunities for compiler optimizations. Profile-based optimization uses trial runs of the code with representative data sets to gather statistics about how the code normally executes. This information can be used to optimize the code to improve performance. By leveraging the PA-8000 core micro-architecture, the PA-8700 protects investment in compilers and application tuning. This enables the compiler to perform optimizations not possible without knowledge of the dynamic behavior of the program. PBO also enables the compiler to more accurately encode static branch prediction hints. The investments made by software developers in PBO will pay off more and more as compilers improve their ability to exploit the increasing potential of future processors.

44-Bit Addressing – Connect to Larger Memory

Previous members of the PA-8x00 family have implemented 40-bits of physical address—enough to address 1 Terabyte of physical memory. While no system that Hewlett-Packard currently ships can hold this much memory, system needs continue to increase, driven by larger data-sets. In the coming years, some customers may have a requirement for more than 1 Terabyte of physical memory. The PA-8700 implements 44-bit physical addresses—enough to address 16 Terabytes of physical memory. The PA-8700 can also operate in 40-bit mode for capability with existing systems.

Data Cache Pre-fetch Capabilities – Data Ready Before It's Needed

Pre-fetching data before it is needed improves the cache hit rate (the probability that the data you want is in the cache when you need it) and thus boosts application performance. The PA-8700 supports pre-fetch capability for the data cache. When a data cache miss occurs, the line that misses is fetched and placed into the data cache. If data pre-fetch is enabled, an adjacent line is also fetched if it is not already in cache. This pre-fetching of a line likely to be used in the near future, allows the line to be brought into the cache pro-actively rather than waiting for a costly cache miss to occur to retrieve the line.

Quasi LRU Cache Replacement Algorithm for Data and Instruction Cache

Complex processors like the PA-8700 need to maximize the probability that the required data is present when an access to the data or instruction cache is made. If the data is not present, then the processor must fetch the needed data from main memory, wasting precious microseconds. The PA-8x00 family incorporates a quasi LRU (Least Recently Used) algorithm to improve the odds that the most frequently used data will remain in the cache.

Whenever new data is moved into a four-way set associative cache, there are four possible locations where the new data can be placed. The replacement algorithm used by the cache determines which of these four locations will have its existing data replaced by the new data. Ideally, the cache would pick the oldest data least likely to be used in the future and replace it with the new data. The PA-8500 uses a 'round-robin' replacement algorithm to determine which data to replace. This replacement selection occurs without any consideration of whether or not a line has been recently used and is thus likely to be used again. An LRU algorithm takes data usage history into account. It identifies which of the four locations in cache was least-recently-used and selects that location for replacement with the new data. In most cases, the data least-recently-used is least likely to be used again in the future and is therefore the best candidate to be replaced with new data being brought into the cache. Unfortunately, a true LRU policy is expensive to implement. Instead, the PA-8700 provides a quasi Least Recently Used (quasi LRU) algorithm. Performance simulations show that the PA-8700 quasi LRU algorithm performs almost as well as a true LRU algorithm while requiring only a fraction of the hardware resources to implement. The use of a quasi LRU replacement algorithm boosts cache hit rates by making better choices about what cache data to replace, and thus allows applications to perform better. The PA-8600 implemented a quasi LRU replacement algorithm for the instruction cache that provides similar advantages for instruction cache performance. The instruction quasi LRU replacement algorithm is also implemented on the PA-8700.

Lock-step Functionality – Hardware-implemented Fault Tolerance

The PA-8700 supports a lock-step mode. In lock-step mode, two or more processors can operate in parallel. On a state-by-state basis, it is possible to compare the results produced by each processor, pin-to-pin. Any discrepancy in results indicates that an error has occurred. By operating two parallel chips in lock-step mode, it is possible to detect a failure by either of the processors. Three parallel chips operating in lock-step mode make it possible to correct a failure by any one of the three processors or detect an error by any two of the processors. This provides the capability to build fault tolerant systems. By paralleling even more chips it is possible to provide even greater levels of fault tolerance. If a discrepancy occurs, it is up to the system to determine what action to take to respond to the fault.

6. PA-8700 Implementation

The PA-8700 represents a continuation of the PA-8000 family evolution. It is implemented in an advanced .18 µm Silicon-on-Insulator (SOI) CMOS process. The process provides 7 layers of copper interconnect with a low epsilon dielectric to minimize RC delay. An additional local interconnect layer is provided for contacting source and drain regions which reduces the need to use metal 1 for this purpose. Table 1 compares the metal interconnect pitches for the PA-8600 process and the PA-8700 process. The advanced process used by the PA-8700 enables it to operate at above 800MHz providing roughly a 50% increase in frequency over the PA-8600. Likewise, the higher level of integration afforded by the .18 µm design rules allows a 50% increase in the size of the on-chip caches compared to the PA-8600. The PA-8700 on-chip instruction cache is .75 Mbytes while the on-chip data cache is 1.5 Mbytes.

Table 1 Metal Pitches

Layer	PA-8600 Pitch	PA-8700 Pitch
Metal 1	.64 µm	.49 µm
Metal 2	.93 µm	.63 µm
Metal 3	.93 µm	.63 µm
Metal 4	1.60 µm	.63 µm
Metal 5	2.56 µm	.63 µm
Metal 6	Not applicable	1.26 µm
Metal 7	Not applicable	1.26 µm

The shorter channels and thinner gate oxides provided by the new process allow a reduction in supply voltage while still providing high performance. The PA-8700 supply voltage is approximately 75% of the supply voltage of the PA-8600. Because power dissipation is proportional to the square of the supply voltage, the reduction in supply voltage reduces power significantly. The lower capacitance provided by the new process further reduces power. The power savings from lower voltage and lower capacitance helps to counteract the increase in power due to higher frequency operation.

Migrating the PA-8700 to an Advanced IC Process

The PA-8700 design team faced a number of challenges moving the circuit to an advanced process. The process used for the PA-8600 was radically different from the standpoint of both geometric artwork rules and electrical performance. The first step in porting the design was to geometrically transform the artwork to conform to the new design rules. This was greatly complicated by the fact that the artwork grid used by the old process was incompatible with the artwork grid used by the new process. In addition, there was not a strict 1-to-1 mapping between the layers of the old and new processes. A program was written to perform this artwork transformation task on a leaf-cell-by-leaf-cell basis. It was recognized early in the project that it would be very difficult to make the program handle every corner case in the transformation process correctly. Instead, the strategy was to construct a program that worked correctly for most circumstances but might generate incorrect artwork for certain hard-to-handle situations. As a result, each block's artwork had to be manually checked by an engineer for problems. This approach worked well because there were numerous electrical issues involved in moving from a bulk CMOS process to an SOI CMOS process, which required intervention by an engineer anyway. The artwork transformation program made its best guess at moving the shapes from the design rules for the old process to the design rules for the new process. An engineer then checked the cell’s artwork in the new process. If problems were discovered, the artwork was fixed by hand. Fortunately, it was very easy to determine when the transformation program generated bad artwork. In most cases the manual fixes were quick and easy to make. Overall, the program did a very good job in transforming the artwork and the engineer checking the new artwork usually had to make a few minor changes, if any.

Once the artwork geometry was ported to the new process, the design team worked on identifying and correcting electrical issues. Although SOI CMOS has a number of advantages compared to a comparable bulk CMOS process, it also has several anomalies that must be compensated for if the circuit is to perform properly. A considerable amount of time was spent at the beginning of the project investigating the electrical properties of the new process. Engineers studied the differences between a bulk and SOI CMOS process. Special consideration was given to the types of circuits that were commonly used on the PA-8600. As a result of this work, guidelines were developed for determining which circuits would be most susceptible to problems in the new process and special situations to watch for. Rules-of-thumb were also developed for designing robust circuits. Finally, simulation techniques were developed to insure that worst-case behavior could be accurately determined for circuits in the new process.

The most serious of the problems presented by the SOI CMOS process were higher leakage current and history dependent changes in a transistor's threshold voltage (V_T). Noise effects are also more of a concern as V_T is now a smaller percentage of the supply voltage. Latches and dynamic circuits are especially susceptible to these problems. Circuits were carefully scrutinized for any potential problems. In many cases, changes had to be made. It was often necessary to increase the size of holder circuits. In many cases, interstitial prechargers (which had been added to counteract charge sharing in bulk CMOS) had to be removed to prevent SOI CMOS leakage problems. Most latches employing NFET-only pass gates were redesigned using fully complementary pass gates. Whenever possible, a latch with unbuffered inputs was replaced with a latch that had buffered inputs to improve its susceptibility to noise.

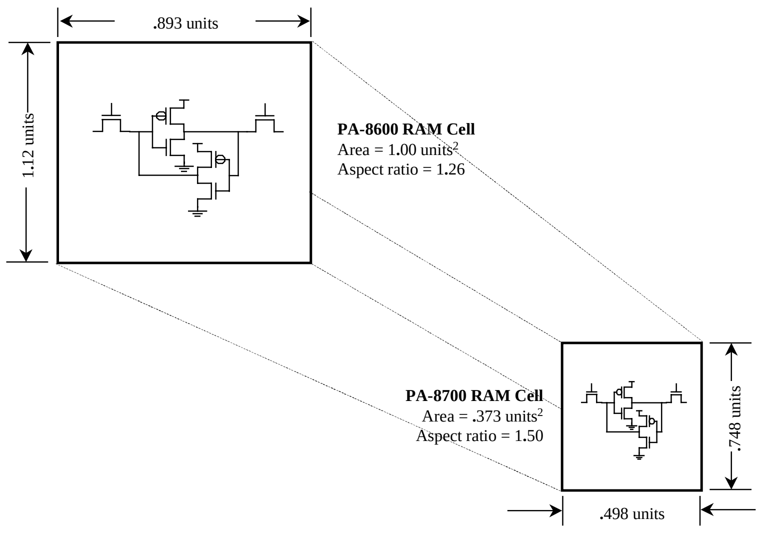
The history dependence of V_T caused serious problems where transistors had to be matched. The only practical solution was to make a connection to the body of the transistor and tie it to a known potential. Such connections caused a significant penalty in both area and speed but did provide predictable and repeatable behavior. Fortunately, this was only required in a few limited situations such as sense amp circuits. The other problem caused by the history dependence of V_T was the uncertainty it introduced in timing. Depending on the past switching history of the transistors in a circuit, delay could vary by up to 10%. Extra analysis was required to make sure that circuits would not experience fatal race conditions due to this added uncertainty in delay through various paths. The simulation techniques developed at the beginning of the program proved particularly important for insuring a race-free circuit behavior.

There were a number of additional changes that had to be made such as removing N_{well} and P_{well} contacts and antenna diodes that were no longer required in the new process. It was also found that many blocks had a problem caused by poor scaling of the metal layers 2 and 3 compared to the previous process. This resulted in excessive RC delays on some signals. The problem was addressed in these blocks by promoting metal 2 and 3 in the old process to metal 4 and 5 in the new process.

PA-8700 Cache Implementation

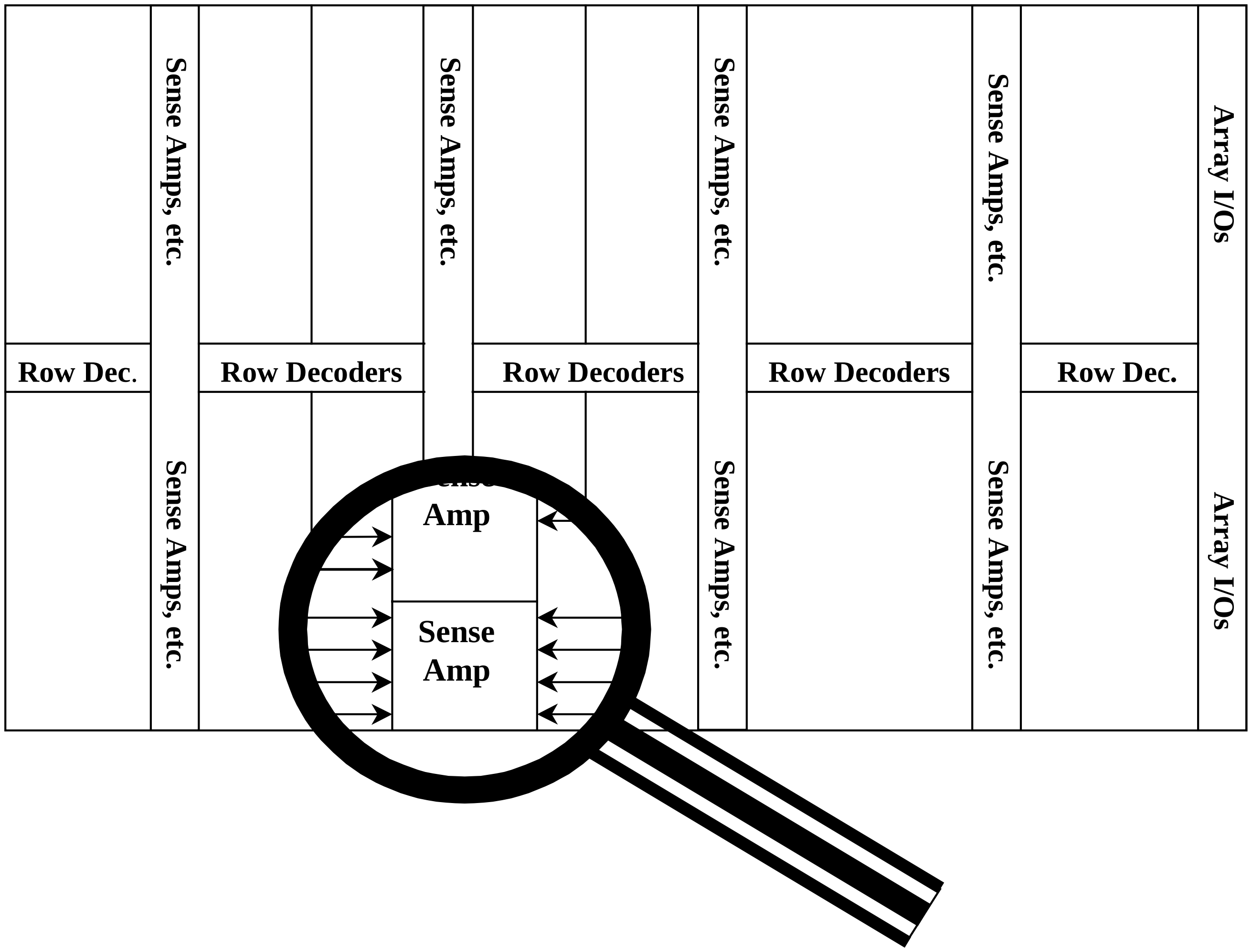
The PA-8700 cache arrays represent the biggest blocks on the chip. There were a number of challenges moving the design of the caches from the PA-8600 to the PA-8700. Due to the large number used, it is critical that the RAM memory cell is highly optimized with respect to the process. For this reason both the PA-8600 and the PA-8700 used RAM memory cell designs provided by the process vendor. Figure 4 shows the transition from the PA-8600 RAM cell to the PA-8700 RAM cell. As would be expected, the area shrank dramatically. A straightforward cell shrink from a .25 µm process to a .18 µm process results in a cell with 52% of the original area. As Figure 4 shows the shrink was actually to 37% of the original area. This demonstrates the excellent packing density provided by the PA-8700 process. Figure 4 also shows that the aspect ratio (height/width) increased from 1.26 on the PA-8600 to 1.50 on the PA-8700. It is interesting to note that on the PA-8700, the total area occupied by the various RAM arrays is 52% of the die area. This is virtually unchanged from the PA-8600 even though the PA-8700 supports 50%larger caches.

Figure 4 RAM Cell Size Transition from PA-8600 to PA-8700 (.25 µm to .18 µm) Process



When the PA-8700 program began, the cache design team investigated what cache sizes could reasonably be provided. Based on the aggressive size of the RAM cell, it was determined that a 50% increase in total cache size was attainable. Alternatives were evaluated to determine how to best provide this increase in size. It was strongly desired that as few of the cache peripheral circuits change as possible. Figure 5 shows the basic architecture of the data arrays for the PA-8600. Four columns of RAM cells on left side and four columns of RAM cells on the right side are multiplexed onto the input of a single sense amp. After examining the alternatives, it was determined that the best way to provide the 50% increase in cache size was to increase the number of columns that are multiplexed onto the input of a single sense amp. The PA-8700 cache team changed the column multiplexer so that 12 columns (6 columns on each side) were multiplexed together instead of the 8 columns in the PA-8600 design. This resulted in a 50% increase in cache size without requiring a redesign of most of the cache peripheral circuits. This also worked very nicely with the new RAM cell's larger aspect ratio.

Figure 5 Cache Array Organization



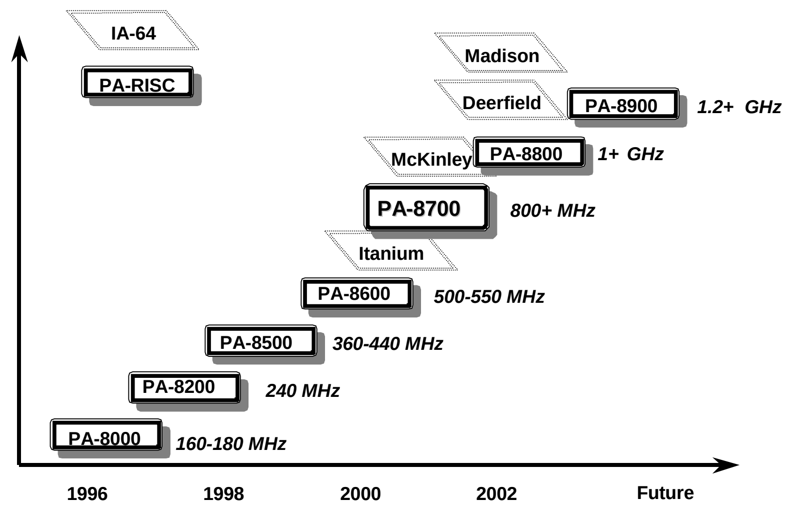
Another issue that the cache design team addressed was increasing chip yield to lower overall part cost. Because the cache arrays occupy a majority of the chip's die area; it was a logical place to look for such opportunities. Memory arrays are regular structures and are thus suitable for adding redundancy to improve manufacturing yield. The PA-8600 added one redundant column to each of the various cache arrays (there are a total of 19 cache arrays, see Figure 3). For each of the arrays, if a column is found to have one or more faults, it can be switched out and replaced with the redundant column. This feature was used very successfully on the PA-8600 to increase chip yield. To improve yield even further, the PA-8700 cache design team increased the number of redundant columns in each array to 4. They also added 32 redundant rows to each cache array to allow swapping out bad rows. The addition of row redundancy to the PA-8700 as well as increasing the number of redundant columns will significantly improve the manufacturing yield and thus help to keep the part cost as low as possible.

7. HP's Microprocessor Roadmap – A Compelling Story

This paper has demonstrated the history of Hewlett-Packard's experience in designing, implementing, and delivering robust computing solutions to our customers using the PA-RISC processor family as a fundamental building block. As Figure 6 illustrates, Hewlett-Packard is fully committed to extensions of the PA-RISC family beyond the PA-8700. In fact, Hewlett-Packard is committed to providing customers a compelling PA-RISC roadmap through the PA-8900.

Hewlett-Packard has a long history of experience in designing, implementing, and delivering robust computing solutions to our customers. The PA-RISC processor family is a fundamental building block in Hewlett-Packard's strategy of delivering robust computing solutions to our customers. Hewlett-Packard is fully committed to extensions of the PA-RISC family beyond the PA-8700, as illustrated in Figure 6. Succeeding PA-RISC microprocessors will easily run at frequencies above 1 GHz.

Figure 6 HP's Microprocessor Roadmap



Hewlett-Packard's CPU roadmap, in Figure 6, includes the IA-64 family of processors beginning with Itanium^{TM} in late 2000 and continuing with McKinley^{TM}, Deerfield^{TM}, and Madison^{TM}. IA-64 is an architecture co-invented by Hewlett-Packard and Intel in a partnership that began in 1994. IA-64 introduces the concept of EPIC (Explicitly Parallel Instruction Computing)–an architecture that exploits instruction-level parallelism – that is, specialized features (predication and speculation) to execute more instructions during each cycle than current generation of out-of-order RISC designs. EPIC computing will address the long-term growth and scalability concerns that RISC-based systems will eventually encounter. IA-64 will provide a pervasive, industry standard 64-bit architecture.

Hewlett-Packard views IA-64 as an evolutionary path from PA-RISC. In many ways, PA-RISC lives on in IA-64. Because of its close association with the development of IA-64, Hewlett-Packard is uniquely positioned to deliver the smoothest possible product transitions for our customers. As Hewlett-Packard deploys IA-64 based systems, customers can expect the following:

- One of Hewlett-Packard’s contributions to the IA-64 effort was the instruction set. A significant number of PA-RISC instructions exist in the IA-64 instruction set. This enables a direct 1-to-1 mapping of machine-level instructions from PA-RISC to IA-64 in most cases. IA-64’s flexibility in handling addresses as either big or little-endian also simplifies the transition from PA-RISC to IA-64. This means that source code binaries originally compiled on PA-RISC can be run very efficiently on IA-64 systems – there is no need to recompile, and the binaries will run on IA-64, unchanged. Another worthy point to note – in addition to full PA-RISC binary availability, HP customers will have data compatibility too. In Hewlett-Packard’s case, there is no requirement to migrate data. These distinct advantages are a direct consequence of the co-development work with Intel on the IA-64 architecture. No other system vendor can claim this binary compatibility.

- Hewlett-Packard will continue to provide PA-RISC based solutions at least through the PA-8900 to ensure our customers have a smooth transition to IA-64.

- Hewlett-Packard will continue to provide product leadership during the transition from PA-RISC to IA-64. Hewlett-Packard understands the fundamentals of the IA-64 architecture, as it was incubated and first developed by HP Labs before collaboration with Intel began. Hewlett-Packard has firsthand knowledge of the strengths of IA-64, and this has translated into unique insight in designing chipsets, systems, compilers, and operating systems. In fact, Hewlett-Packard was the first vendor to release an IA-64 upgradable system in April 1999–the N-class server.

- Hewlett-Packard will be the solution vendor with the largest set of applications that will run on IA-64. HP systems will support the thousands of applications that currently run on HP-UX, Linux, and Windows NT64. Hewlett-Packard will have a comprehensive group of Independent Software Vendors (ISVs) that will migrate their applications from PA-RISC to IA-64. Running applications natively on IA-64 will take full advantage of IA-64’s EPIC technology.

Hewlett-Packard’s strategy benefits our customers by allowing them to preserve their current infrastructure investment (in PA-RISC), while allowing them the upgrade to the future (in IA-64). Hewlett-Packard will offer customers the choice of when they migrate from PA-RISC to IA-64. This ability to choose, along with its long track record for delivering leadership products, distinguishes Hewlett-Packard from its competition. Hewlett-Packard will be the leader in IA-64 solutions.