PowerPC Architecture Gets Makeover

"Book E" Adds New Features for Embedded Market

by Keith Diefendorff

Showing some sign of improved relations, IBM and Motorola have completed development of a new version of the PowerPC instruction-set architecture, code-named Book E. At last week's Embedded Processor Forum, IBM chip architect Tom Sartorius said that the new architecture will allow PowerPC chips to better meet the needs of embedded markets while retaining backward compatibility with existing 32-bit user-mode PowerPC applications. With this move, PowerPC officially joins the ranks of other RISC processors seeking shelter in the embedded market, safe for now from their omnipotent competitor, Intel.

EMBEDDED PROCESSOR FORUM

As Table 1 shows, Book E offers enhanced 64-bit addressing, simplified memory management, and an interrupt structure suitable for real-time applications. Many of the changes from the original PowerPC architecture were made to give the companies more freedom to customize their processors, a critical requirement for embedded markets. Unlike the original PowerPC architecture, the new architecture will be freely licensable and sublicensable by either company.

IBM chip architect Tom Sartorius describes the new Book E architecture developed with Motorola.

The specification of an architecture is a tradeoff. On the one hand, a rigid specification promotes software compatibility. On the other hand, a flexible specification allows customization and differentiation. Initially targeted for computers, where compatibility reigns, the original PowerPC specification is relatively rigid. For embedded duty, where customization is paramount, IBM and Motorola needed a more flexible version of the architecture. Although Book E standardizes a few of the trivial differences between IBM's 4xx and Motorola's 5xx/8xx embedded processors, it opens wide the door to software incompatibility. Under Book E, avoiding architectural chaos will depend heavily on the self-discipline of the two companies—a commodity often sacrificed to gain a competitive edge.

Controllers Need 64 Bits Too

Although the need for 64-bit addressing is well recognized to be just around the corner for computer systems, it is generally regarded as over the horizon for embedded systems. Motorola architect Pete Wilson, however, says that 64-bit addressing was the primary impetus for Book E. According to Wilson, several embedded applications are beginning to feel the pinch of 32-bit addresses; a RAID controller, for example, will soon need more than 4G of memory just to hold the tables for tracking data in a terabyte-sized storage system. Large switches, already using 1G of memory, will exceed 4G within a couple of years. Wilson adds that patchwork physical-address extensions, like Pentium II's IESM (see MPR 7/13/98, p. 7), won't cut it—programmers need full 64-bit pointers.

Although the original PowerPC has 64-bit addressing, that feature was designed primarily for AIX. The mode-switching technique it uses to support 32- and 64-bit implementations has the disadvantage that a 32-bit binary will not run on a 64-bit processor unless it is first switched into 32-bit mode. While mode switching isn't a problem for AIX, it is a heavy burden for porting the plethora of embedded OSs to PowerPC.

Unconstrained by AIX in embedded applications, the companies were able to eliminate the cumbersome 32/64-bit mode switch, which they accomplished without sacrificing the 32-bit application-level binary compatibility necessary to sell Book E processors to Apple. So Book E leaves the original PowerPC 64-bit extension as the sole province of IBM's Power series of servers and workstations.

Mode Traded for Instructions

In place of the existing 32/64-bit mode switch, Book E clones the carry flag, overflow flag, and all mode-dependent instructions, creating 32- and 64-bit versions of each.

On an original 64-bit PowerPC machine, for example, the ADDC (add carrying) instruction sets the carry flag (CA) from the lower half of the 64-bit ALU in 32-bit mode and

Feature	Original PowerPC	PowerPC Book E
64-bit Addressing	Modal	Nonmodal, 100 new instructions
Memory Management	Hashed page tables	TLB-centric, implementation defined
Little-Endian Byte Order	Address swizzle	Byte swapping (true little-endian)
Memory Ordering	Weak; processor sync	Weak; memory barrier
Interrupts	Single level	Two level, critical and noncritical
Interrupt Vectors	Fixed; in physical memory	Programmable; in virtual memory
Timers	Decrementer, timebase	+ fixed-interval timer and watchdog
New Opcode Assignment	Apple & IBM & Motorola	IBM or Motorola

Table 1. Book E's new features improve PowerPC's suitability for embedded processors without sacrificing 32-bit user-mode compatibility with PowerPC-classic processors.

from the full ALU in 64-bit mode; ADDE (add extended) always uses CA as its input. In contrast, the Book E version of ADDC sets two flags: CA and CA64. The ADDE instruction adds using CA while a new instruction, ADDE64, uses CA64. Most instructions, like normal ADD, do not require both forms, because their results are indistinguishable by a 32-bit program running on either a 32- or 64-bit machine.

The Book E 64-bit design, although conceptually simple, requires 94 more instructions than the original design: 124 new instructions were added; 30 were eliminated. To accommodate the new instructions within the opcode space, Book E sacrifices four bits of displacement in its extended load and store instructions, reducing the reach of the displacement from ±32K to ±2K. Sartorius says most displacements are smaller than 2K anyway, so he expects little performance loss.

The new instructions forced a reassignment of some of the 64-bit-instruction opcodes, severing compatibility with the original 64-bit PowerPC and with IBM's POWER architecture. But Sartorius claims that reassigning opcodes avoided a cycle-time penalty for decoding the extra instructions (compared with an original 64-bit PowerPC implementation).

Address Translation Gets Uninverted

Having cracked open the architecture to simplify 64-bit addressing, the companies took the opportunity to make other changes as well. Chief among them is restructuring memory management from a memory-based page-table design to a TLB-centric design. Actually, Book E nearly eliminates memory management from the specification altogether, leaving much of it to the implementation. Only a few abstract requirements are specified, along with some generic TLB-management instructions.

The address-translation mechanism in the original PowerPC consists of a segment-register-based translation to a 52-bit virtual-address space, followed by a hashed-page-table (a type of inverted-page-table) translation to a physical address. The system has many technical advantages, such as simple data sharing and small page tables (their size is proportional to physical memory, not the virtual-address space as with traditional page tables).

For embedded systems, however, this scheme is too complex, especially for 64-bit machines, which require a segment lookaside buffer (SLB) instead of simple segment registers. (The complexity of the 64-bit memory management was a contributing factor to the PowerPC 620 debacle.)

Complexity aside, the real problem with the original scheme is that the only OS on the planet that uses it is AIX; MacOS, Windows NT, Linux, and all embedded real-time OSs can't use it. Although the scheme works well for an OS designed for it from scratch, it just gets in the way of others. All OSs ever ported to PowerPC (besides AIX) waste time emulating their conventional page tables on top of PowerPC's inverted page tables, sacrificing performance. Book E supports virtually any type of page table, at the discretion of the OS.

Since address translation is not generally visible from user mode, it can be changed without violating application-level compatibility. Changes affect operating system code, but most modern kernel-based OSs are modular enough to accommodate the changes. Even MacOS 8.6 and the future MacOS X should have no trouble adapting. Indeed, not having to cope with inverted page tables will be a blessing.

Feature	Bits	Description
Address Translation (effective address to physical address)		
RPN	≤54	Real page number (variable page size)
TS	1	Translation address space (application or system)
SIZE	4	Page size = 1,024 × 4^{SIZE} (1 KB to 1 TB)
PID	–	Process ID (size is implementation specific)
Access Control (per page)		
UX	1	User-mode execute enable
SX	1	Supervisor-mode execute enable
UW	1	User-mode write enable
SW	1	Supervisor-mode write enable
UR	1	User-mode read enable
SR	1	Supervisor-mode read enable
Storage Attributes (per page)		
WT	1	Write through (writes to cache go to memory)
I	1	Cache inhibited (all accesses bypass the cache)
M	1	Coherent (accesses maintained cache coherent)
G	1	Guarded (accesses performed in program order)
E	1	Endianness (big- or little-endian byte ordering)
U_{0-3}	4	User-defined storage attributes

Table 2. Book E specifies a number of memory-management features that a processor must implement, but it doesn't dictate TLB or memory-based page-table formats.

Abstract Definition Leaves Room for Differences

Although many differences between address-translation mechanisms are easy for an OS to handle, others are not. For example, the bit layout of a page descriptor is usually inconsequential, but differences in page sizes are more problematic, and the absence of write protection could be catastrophic. To avoid this problem yet still gain flexibility vis-à-vis the original PowerPC's explicit specification, Book E defines a required set of features but leaves open the low-level implementation details and how the OS invokes them.

Gone is the specification of any particular memory-based page-table structure. Instead, Book E defines at an abstract level the TLB features that must be implemented, as Table 2 shows. The management of TLB entries and the structure of memory-based mapping tables are left to software. TLB organization (e.g., number of entries and associativity) is left to the implementation. The only specified hardware is the interrupt vectors for TLB exceptions, a register to capture the address of a TLB miss, and instructions for reading, writing, searching, and synchronizing the TLB.

For the most part, storage-access privileges (read and write permissions) and storage attributes (e.g., write through, cache inhibited, cache coherent, and ordered) are the same as those in the original PowerPC. The address-aliasing function of segment registers is replaced by a simple process-ID register (PID). Block-address-translation registers (BATs) and real-address mode have been eliminated, replaced by variable-sized pages ranging from 1KB to 1TB (one terabyte). Four new user-definable storage attributes are available for use by the OS to denote, for example, Java-translated or compressed pages. Other new features include per-page execute protection—previously available only on a segment basis—and per-page byte-order (endianness) selection.

Like existing PowerPC processors, Book E processors will have both big- and little-endian byte ordering. But because little-endian support was added to PowerPC just as the first processor (PowerPC 601) was nearing tapeout, only the simple address-swizzle form of little-endian could be accommodated. This method, adequate for homogeneous systems, is cumbersome in heterogeneous systems.

Thus, Book E adopts true little-endian ordering, implemented by byte swapping. Although this change violates the objective of strict user-mode compatibility, in practice the change should be relatively benign. The difference between address swizzling and byte swapping is often undetectable by programs, and few customers are using the feature anyway.

Book E preserves PowerPC's weak-memory-ordering model, adding a couple of new memory-barrier instructions to reduce the overhead of the software locks and semaphores that are important in multiprocessor systems and multithreaded environments like Java. The original PowerPC actually provides stronger synchronization than is needed for software signaling. Thus, the SYNC instruction, which stalls the execution pipeline, is replaced by a less drastic MSYNC, and MBAR puts EIEIO out to pasture. The new primitives provide only the minimum barriers necessary to ensure that memory accesses made prior to a lock claim (or lock release) are completed before the accesses following it are started.

Two Interrupt Levels Improve Response Time

Under the existing single-level structure, interrupts cannot be nested until the first-level handler can save enough machine state to safely reenable interrupts. The time period during which interrupts must be disabled can exceed the response-time requirements of some real-time systems.

To correct this shortcoming, Book E defines a "critical" interrupt level that can be taken at any time during the processing of "noncritical" interrupts. Noncritical interrupts signal ordinary events such as memory-protection violations, illegal opcodes, internal-timer events, and external asynchronous inputs. Critical interrupts signal higher-priority events, including debug traps, critical external asynchronous inputs, machine checks, and watchdog-timer events.

Other changes further increase the power and flexibility of the interrupt system to take on embedded-system duty. Interrupt vectors, for example, were previously hardwired to fixed addresses in physical memory, but now they have been made programmable in both location and size and are mapped in virtual memory. PowerPC's timer facilities were beefed up by adding an automatic-reload feature to the decrementer and fixed-interval triggers to the time-base counter. These features reduce software overhead for handling trivial timer events. Unlike in the original PowerPC, timer interrupts can be disabled without disabling all interrupts.

A useful new feature is the watchdog timer, a foolproof mechanism for guarding against software errors that can trap a processor in an infinite loop with interrupts disabled. The watchdog is triggered by selected bit transitions of the time-base counter, and it is normally reset by software on each fixed-interval timer event. If the watchdog gets set, indicating a possible problem, a critical interrupt is signaled. If the event handler cannot clear the problem and reset the watchdog, the next watchdog event will force an unmaskable system reset. To assure that nefarious software cannot sabotage the processor, the watchdog is unstoppable: once enabled, it can be disabled only by a system reset.

Book E also defines an extensive set of on-chip debug facilities, including breakpoint events on instruction and data addresses, data values, trap instructions, taken branches, exceptions, and subroutine returns. The debugger specification, however, takes the form of recommended features rather than strict requirements. Specific implementations are free to create a superset or subset of the debug features.

Opcode Flexibility Imperils Compatibility

According to Sartorius, an important motivation for Book E was to gain opcode flexibility. Designed for strict binary

Feature	Size	Description
ADDE64[O][.]	64	Add extended w/CA64 [overflow][cond]
ADDME64[O][.]	64	Add to -1 extended w/CA64
ADDZE64[O][.]	64	Add to zero extended w/CA64
SUBFE64[O][.]	64	Subtract from extended w/CA64
SUBFME64[O][.]	64	Subtract from -1 extended w/CA64
SUBFZE64[O][.]	64	Subtract from zero extended w/CA64
BCCTRE[L]	64	Branch conditional to count reg [link]
BCLRE[L]	64	Branch conditional to link register
BE[L][A]	64	Branch [link] [absolute]
BCE[L][A]	64	Branch conditional
DCB[…]E	64	Data-cache block instructions (7 instrs)
ICBIE	64	Instruction-cache block invalidate
ICBT	32	Instruction-cache block touch
ICBTE	64	Instruction-cache block touch
Loads…	64	Duplicates of original PPC loads (27)
Stores…	64	Duplicates of original PPC stores (24)
MBAR	32/64	Memory barrier
MSYNC	32/64	Memory synchronize
MCRXR64	64	Move to condition register from XER64
MFAPIDI	32/64	Move from APU ID register indirect
M[F/T]DCR	32/64	Move from/to device control register
RFCI	32/64	Return from critical interrupt
TLBIVAX	32	TLB invalidate
TLBIVAEX	64	TLB invalidate
TLBRE	32/64	TLB read entry
TLBWE	32/64	TLB write entry
TLBSX[.]	32	TLB search
TLBSXE[.]	64	TLB search
WRTEE	32/64	Write external interrupt enable
WRTEEI	32/64	Write external interrupt enable immed

Table 3. A total of 124 new instructions are added by Book E, most of them to support the new modeless 64-bit addressing model.

For More Information

For more information on Book E, visit *www.chips.ibm.com/products/powerpc* or *http://motorola.com/PowerPC*. Neither IBM nor Motorola has announced any processors based on Book E.

compatibility, the original PowerPC has a tightly controlled opcode space, requiring unanimous consent by Apple, IBM, and Motorola to modify or extend it. The addition of new instructions, however, is an important method of customizing processors, lack of which would have presented a barrier to serving future embedded markets. Since IBM and Motorola wish to serve different embedded markets, and since Apple's needs are computer focused, a "new" embedded architecture was the path of least resistance around the barrier.

To gain flexibility without totally abandoning compatibility, the Book E opcode space was divided into four categories of instructions: defined, preserved, reserved, and allocated. Defined instructions make up the base instruction set; they include the original 32-bit PowerPC instructions plus the new instructions listed in Table 3. Preserved opcodes are PowerPC supervisor-mode instructions that remain defined for backward compatibility but that may eventually be expunged. Reserved opcodes are set aside for future expansion; their use must be blessed by both IBM and Motorola. (Apple has no official say in Book E.)

Allocated instructions—which include two full primary opcodes (4,096 potential instructions) plus 512 secondary opcodes (in four other primary opcodes)—are reserved as "sandbox" opcodes. Either company is free to play in this sandbox, assigning opcodes to any purpose it sees fit, including new user-mode instructions. Therein lies the risk: each company could, in theory, elect to use the same opcode for different instructions, leading to incompatible chips. Although this eventuality is not precluded, the companies expect it will not be a problem in embedded markets, which often don't overlap in software. Other architectures, like MIPS, have thrived with similar characteristics.

As a method of officially supporting new functions, Book E introduces the notion of an application-specific processing unit (APU) similar in concept to the MIPS coprocessor. Architecturally, APUs are little more than a convenient way to talk about facilities for executing allocated instructions; the only defined support is some interrupt vectors for exceptions and an instruction to read APU status.

APUs can be either simple or complex and can share the general-purpose registers or implement a separate register file having its own loads and stores. The integer multiply-add feature in IBM's 4xx processors is an example of a simple APU; AltiVec, the SIMD multimedia extension in Motorola's G4 (see MPR 11/16/98, p. 17), is an example of a complex APU.

Apple can, in theory, use Book E processors in Macintoshes with a few OS changes. Under Book E, however, IBM and Motorola could deliver incompatible processors. Motorola's AltiVec APU, for example, could easily be torpedoed if IBM were to misuse one of its opcodes. Although such activity would not be in Apple's best interests, it is sanctioned by Book E, and it will be tempting for IBM or Motorola to use this capability as a weapon to gain an advantage over the other.

Contrary to earlier rumors, Book E does not define a 16-bit Thumb-like instruction set for reduced code size. Both IBM and Motorola agree that code-compression techniques, like IBM's CodePack (see MPR 10/26/98, p. 26), are just as effective at reducing code size and create fewer compatibility hassles. Therefore, Book E establishes no code-compression standard, leaving it up to individual implementors.

Book E Shifts Emphasis to Embedded

Book E is not as aggressive as it could have been. Although it fixes some annoying problems that PowerPC inherited from IBM's POWER architecture, like inverted page tables, it does not fix all known shortcomings. Several implementation complexities, like string instructions, remain. Fixing these, however, would have required breaking backward compatibility—a move that would have stranded Apple with the original PowerPC and might have pushed it over the edge into the Intel camp.

Although IBM and Motorola point to technical reasons behind the move to Book E, in reality its changes are modest and could easily have been made within the framework of the existing PowerPC architecture, without a grandiose "new" architecture. Thus, Book E appears to be motivated by legal and business considerations as well as technical ones.

Under their original PowerPC contracts, the two companies were overly constrained. Unable to agree on a common direction and not free to mutate PowerPC into the different forms each company needed, the partnership unraveled (see MPR 6/22/98, p. 10). By creating a new architecture, the companies have escaped the legal contracts governing their actions with respect to the original PowerPC. Under those agreements, for example, "PowerPC" was owned by IBM, preventing Motorola from licensing it to others. Under the Book E agreements, Motorola has gained these rights—a benefit as significant as any technical improvement.

IBM and Motorola seem to have learned a lesson from their earlier failed partnership. The duo seems now to recognize what they failed to recognize initially: they are really competitors—a fact that doomed Somerset from the start. Both companies clearly understand that Book E is a precompetitive agreement and, once it is complete, no holds are barred.

On the one hand, Book E weakens the PowerPC standard, opening the door wide to instruction-set differences that threaten software compatibility. On the other hand, since PowerPC has clearly lost its bid to topple the x86 off the desktop, or even to capture a significant share of the PC market—not withstanding iMac—the flexibility offered by Book E may in the long run prove more important to IBM, Motorola, and PowerPC than software compatibility. M