TABLE 6.4 Little-Endian vs. Big-Endian

Value	Endianness	ADDR+0	ADDR+ 1	ADDR+2	ADDR+3
0x1234	Little	0x34	0x12	X	x
0x1234	Big	0x12	0x34	X	x
0x12345678	Little	0x78	0x56	0x34	0x12
0x12345678	Big	0x12	0x34	0x56	0x78

memory as you would read and interpret it. The choice of "endianness" is rather religious and comes down to personal preference. Of course, if you are designing with a little-endian microprocessor, life will be made simpler to maintain the endianness consistently throughout the system.

At the time of the 8086's introduction, 16-bit desktop computer systems were almost unheard of and could be substantially more expensive than 8-bit systems as a result of the increased memory size required to support the larger bus. To alleviate this problem and speed market acceptance of its architecture, Intel introduced the 8088 microprocessor in 1979, which was essentially an 8086 with an eight-bit data bus. A lower-cost computer system could be built with the 8088, because fewer EPROM and RAM chips were necessary, system logic did not have to deal with two bytes at a time, and less circuit board wiring was required. A tremendous benefit to Intel in designing the 8088 was the fact that it was chosen by IBM as the low-cost 16-bit heart of the original PC/XT desktop computer, thereby locking the x86 microprocessor family into the IBM PC architecture for decades to come.

A variety of companion chips were developed by Intel to supplement the 8086/8088. Among these was the 8087 math coprocessor that enhanced the 8086's computational capabilities with *floating-point* arithmetic operations. Floating-point arithmetic refers to a computer's handling of real numbers as compared to integers. The task of adding or multiplying two real numbers of arbitrary magnitude is far more complex than similar integer operations. Certain applications such as scientific simulations and realistic games that construct a virtual reality world make significant use of floating-point operations. The 8087 is a *coprocessor* rather than a peripheral, because it sits on the microprocessor bus in parallel with the 8086 and watches for special floating-point instructions. These instructions are then executed automatically by the 8087 rather than having to wait for the 8086 to request an operation. The 8086 was designed with the 8087's existence in mind and ignores instructions destined for the 8087. Therefore, software must specifically know if a math coprocessor is installed to run correctly. Many programs that ran on older systems with or without a coprocessor would first test to see if the coprocessor was installed and then execute either an optimized set of routines for the 8087 or a slower set of routines that emulated the floating-point operations via conventional 8086 instructions.

As the x86 family developed, the optional math coprocessor was eventually integrated alongside the integer processor on the same silicon chip. The 8087 gave way to the 80287 and 80387 when the 80286 and 80386 microprocessors were produced. When Intel introduced the 80486, the coprocessor, or *floating-point unit* (FPU), was integrated on chip. This integration resulted in a somewhat more expensive product, so Intel released a lower-cost 80486SX microprocessor without the coprocessor. An 80487SX was made available to upgrade systems originally sold with the 80486SX chips, but the overall situation proved somewhat chaotic with various permutations of microprocessors and systems with and without coprocessors. Starting with the Pentium, all of Intel's high-end microprocessors contain integrated FPUs. This trend is not unique to Intel. High-performance microprocessors in general began integrating the FPU at roughly the same time because of the performance benefits and the overall simplicity of placing the microprocessor and FPU onto the same chip.

6.6 MOTOROLA 68000 16/32-BIT MICROPROCESSOR FAMILY

Motorola followed its 6800 family by leaping directly to a hybrid 16/32-bit microprocessor architecture. Introduced in 1979, the 68000 is a 16-bit microprocessor, due to its 16-bit ALU, but it contains all 32-bit registers and a linear, nonsegmented 32-bit address space. (The original 68000 did not bring out all 32 address bits as signal pins but, more importantly, there are no architectural limitations of using all 32 bits.) That the register and memory architecture is inherently 32 bits made the 68000 family easily scalable to a full 32-bit internal architecture. Motorola upgraded the 68000 family with true 32-bit devices, including the 68020, 68040, and 68060, until switching to the PowerPC architecture in the latter portion of the 1990s for new high-performance computing applications. Apple Computer used the 68000 family in their popular line of Macintosh desktop computers. Today, the 68000 family lives on primarily as a mid-level embedded-processor core product. Motorola manufacturers a variety of high-end microcontrollers that use 32-bit 68000 microprocessor cores. However, in recent years Motorola has begun migrating these products, as well as their general-purpose microprocessors, to the PowerPC architecture, reducing the number of new designs that use the 68000 family.

The 68000 inherently supports modern software *operating systems* (OSs) by recognizing two modes of operation: supervisor mode and user mode. A modern OS does not grant unlimited access to application software in using the computer's resources. Rather, the OS establishes a restricted operating environment into which a program is loaded. Depending on the specific OS, applications may not be able to access certain areas of memory or I/O devices that have been declared off limits by the OS. This can prevent a fault in one program from crashing the entire computer system. The OS *kernel*, the core low-level software that keeps the computer running properly, has special privileges that allow it unrestricted access to the computer for the purposes of establishing all of the rules and boundaries under which programs run. Hardware support for multiple privilege levels is crucial for such a scheme to prevent unauthorized programs from freely accessing restricted resources. As microprocessors developed over the last few decades, more hardware support for OS privileges was added. That the 68000 included such concepts in 1979 is a testimony to its scalable architecture.

Sixteen 32-bit general-purpose registers, one of which is a user stack pointer (USP), and an 8-bit condition code register are accessible from user mode as shown in Fig. 6.11. Additionally, a supervisor stack pointer (SSP) and eight additional status bits are accessible from supervisor mode. Computer systems do not have to implement the two modes of operation if the application does not require it. In such cases, the 68000 can be run permanently in supervisor mode to enable full access to all resources by all programs. The SSP is used for stack operations while in supervisor mode, and the USP is used for stack operations in user mode. User mode programs cannot change the USP, preventing them from relocating their stacks. Most modern operating systems are *multitasking*, meaning that they run multiple programs simultaneously. In reality, a microprocessor can only run one program at a time. A multitasking OS uses a timer to periodically interrupt the microprocessor, perhaps 20 to 100 times per second, and place it into supervisor mode. Each time supervisor mode is invoked, the kernel performs various maintenance tasks and swaps the currently running program with the next program in the list of running programs. This swap, or *context switch*, can entail substantial modifications to the microprocessor's state when it returns from the kernel timer interrupt. In the case of an original 68000 microprocessor, the kernel could change the return value of the PC, USP, the 16 general-purpose registers, and the status register. When normal execution resumes, the microprocessor is now executing a different program in exactly the same state at which it was previously interrupted, because all of its registers are in the same state in which they were left. In such a scenario, each program has its own private stack, pointed to by a kernel-designated stack pointer.

The eight data registers, D0–D7, can be used for arbitrary ALU operations. The eight address registers, A0–A7, can all be used as base addresses for indirect addressing and for certain 16- and 32-bit