Letters

To the Editor

More on big-endian vs. little-endian byte ordering

To the Editor:

I prefer a different summary of my big-endian versus little-endian views than that offered by Tim Paterson (Letters, IEEE Micro, February 1984, page 11).

Let us use the terminology “high-first” and “low-first,” instead of “big-endian” and “little-endian,” to reduce the emotional content and clarify the meaning. “High-first” means that the most significant (high) byte of an integer is stored at the lowest (first) address.

The arguments favoring high-first byte ordering and those favoring low-first byte ordering cancel each other, when properly paired, except for one: Character strings are addressed most-significant-byte-first in both schemes, but multibyte integers or addresses are addressed least-significant-byte-first in the low-first one.

This low-first asymmetry is the cause of most of the inconveniences which plague users of that scheme. It also causes most of the difficulty in interfacing the two schemes, because it creates a need for context-dependent byte swapping. Since context information (the “meaning” of the bytes) is not usually available to hardware or to general software utilities, transforming data from one scheme to the other is intrinsically difficult and requires ad hoc algorithms for every application.

The inconsistent numbering of bits and bytes often used in today’s high-first microprocessors is inelegant; however, in practice the bit numbering is orders of magnitude less important than the byte-addressing problem, which causes significant wasted time for both humans and computers. This is because the significance of bits within bytes is the same in all contexts—only the bit number labels differ between the two systems, and in practice they are rarely used in a way that affects data portability.

New designs ought to use the high-first scheme for both byte addressing and bit numbering. However, it is not a serious problem if they use the low-first scheme for bit numbering, as the MC68000 and Z8000 do. By adopting the main virtue of the low-first scheme, these practical hybrids have the best of both, paying only a minor penalty for their slight inconsistency.

Of course, we must forgive those designers who, because of compatibility considerations, are constrained to the optimizations and errors of the past. That explains the 8086, which is an enhanced 8080, which is an enhanced 8008, whose address arithmetic was simplified by storing least significant bytes first in memory. But how sad that the otherwise elegant PDP-11 and the NS16000 adopted the internally inconsistent scheme.

David B. Gustavson
Stanford Linear Accelerator Center
PO Box 4349, Bin 88
Stanford, CA 94305

Semaphore strategy for Z80

To the Editor:

In his article, “A System Executive for Real-Time Microcomputer Programs” (IEEE Micro, June 1984, pp. 20-32), Walter S. Heath implies that a semaphore cannot be directly supported by the Z80 microprocessor because of the lack of a test-and-set instruction. The solution proposed is to simulate the test and set by first disabling the interrupts. This is a common criticism of many eight-bit micros. However, in many cases, including the Z80, there exists instead a set-and-test mechanism which allows direct implementation of a semaphore without circuitous programming.

This mechanism is the shift in memory. My own preference is the shift right logical in memory (Z80 mnemonic SRL). Utilizing this instruction, one can support a semaphore through the following procedure:

1. Initialize the memory semaphore by setting only its least significant bit (LSB).
2. To set the semaphore perform the SRL. This shifts the LSB into the C flag in a single indivisible operation.
3. Test the C flag. If set, the semaphore is open. If clear, the semaphore is locked.
4. If open, proceed to use the now-protected resource. When finished, unlock the resource by writing a one to the semaphore location.

Since operation two is indivisible, it effectively locks out any subsequent or interrupting task seeking access to a protected resource.

Wolfram Lunscher
Research & Development
Develcon Electronics Ltd.