Which bit should travel first? The bit from the big end or the bit from the little end? Can a war between Big Endians and Little Endians be avoided?

On Holy Wars and

a Plea for Peace

Danny Cohen
Information Sciences Institute

This article was written in an attempt to stop a war. I hope it is not too late for peace to prevail again. Many believe that the central question of this war is, What is the proper byte order in messages? More specifically, the question is, Which bit should travel first—the bit from the little end of the word or the bit from the big end of the word?

Followers of the former approach are called Little Endians, or Lilliputians; followers of the latter are called Big Endians, or Blefuscuians. I employ these Swiftian terms because this modern conflict is so reminiscent of the holy war described in *Gulliver's Travels*.[1]

Approaches to serialization

The above question arises as a result of the serialization process performed on messages to allow them to be sent through communication media. If the unit of communication is a message, this question has no meaning. If the units are computer words, one must determine their size and the order in which they are sent.

Since they are sent virtually at once, there is no need to determine the order of the elements of these words.

If the unit of transmission is an eight-bit byte, questions about bytes are meaningful but questions about the order of the elementary particles that constitute these bytes are not.

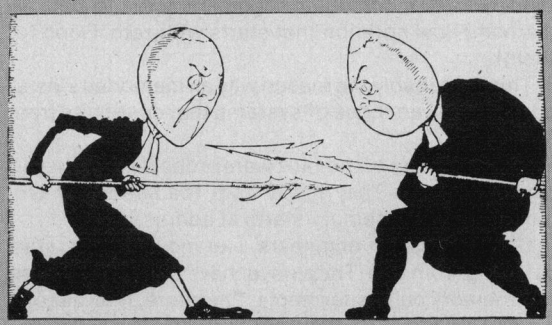
If the units of communication are bits, the atoms (quarks?) of computation, the only meaningful question concerns the order in which the bits are sent. Most modern communication is based on a single stream of information, the bit-stream. Hence, bits, rather than bytes or words, are the units of information that are actually transmitted over channels such as wires and satellites.

Notes on Swift's *Gulliver's Travels*

Swift's hero, Gulliver, is shipwrecked and washed ashore on Lilliput, whose six-inch inhabitants are required by law to break their eggs only at the little ends. Of course, all those citizens who habitually break their eggs at the big ends are angered by the proclamation. Civil war breaks out between the Little Endians and the Big Endians, resulting in the Big Endians taking refuge on a nearby island, the kingdom of Blefuscu. The controversy is ethically and politically important for the Lilliputians. In fact, Swift has 11,000 Lilliputian rebels die over the egg question. The issue might seem silly, but Swift is satirizing the actual causes of religious or holy wars.

Swift's point is that the difference between breaking an egg at the little end and breaking it at the big end is trivial. He suggests that everyone do it in his preferred way.

Of course, we are making the opposite point. We agree that the difference between sending information with the little or the big end first is trivial, but insist that everyone must do it in the same way to avoid anarchy.



Reproduced from *The Annotated Gulliver's Travels* by Isaac Asimov. Published by Crown Publishers, Inc.

Facing page: "Audience with the Emperor of Lilliput," Gavarni, 1850's. Courtesy Library of Congress.

0018-9162/81/1000-0048$00.75 © 1981 IEEE

Even though a great deal of effort in both hardware and software is dedicated to giving the appearance of byte or word communication, the fact is that only bits are communicated.

Computer memory can be considered a linear sequence of bits organized into bytes, words, pages, and so on. Each unit is a subunit of the next higher level. This is, obviously, a hierarchical organization.

If the order is consistent, all parties to the communication can treat the bits as a set of groups of any arbitrary size. One party might treat a message as a page while another might treat it as so many words, so many bytes, or so many bits. If a consistent bit order is used, the chunk size is of no consequence.

If the bit order is inconsistent, the chunk size must be understood and agreed upon by all parties. We shall demonstrate some popular but inconsistent orders later.

Consistent order vs. inconsistent order. A consistent order is one in which the bit, byte, word, page, and all higher-level orders are the same. Hence, the originator's concept of chunk size is unimportant in consideration of a serial bit stream along a communication line.

There are two possible consistent orders. One starts with the narrow end (LSB) of each word favored by the Little Endians; the other starts with the wide end (MSB) preferred by the Big Endians.

In the discussions to follow, these definitions and abbreviations apply: a *word* is a 32-bit quantity designated by a W; a *byte* is an eight-bit quantity designated by a C (for *character*); and a B stands for *bit*.

Memory order

The first word in memory is designated W0 in both regimes. Unfortunately, the harmony goes no further. The Little Endians assign B0 to the LSB of the words and B31 to the MSB. The Big Endians do just the opposite: B0 is the MSB and B31 is the LSB.

By the way, if mathematicians had their way, every sequence would be numbered from zero, not from the traditional one. Under this scheme, the first item would be called the *zeroth*. Since most computers are not built by mathematicians, it is no wonder that some computers designate bits from B1 to B32 in both the Little Endians' and the Big Endians' order. Designers of such machines probably would like to number their words from W1 up, just to be consistent.

To illustrate the hierarchically consistent order graphically, one must first decide the order in which to write computer words on paper. Do they go from left to right, or from right to left?

English, like most modern languages, suggests that we arrange computer words from left to right:

```
|---word0---|---word1---|---word2---|....
```

To be consistent, B0 should be to the left of B31. If the bytes in a word are designated as C0 through C3, then C0 should be to the left of C3:

```
|---word0---|---word1---|---word2---|....
|C0,C1,C2,C3|C0,C1,C2,C3|C0,C1,C2,C3|.....
|B0.....B31|B0.....B31|B0.....B31|......
```

The convention introduced by our numbering system places the wide end on the left and the narrow end on the right.

The above is a perfectly consistent view of the world according to the Big Endians. The significance decreases with increasing item numbers (address).

Many computers share the Big Endians' point of view. In many Big Endian diagrams, the registers are connected

Who's on first? Zero or one?

People start counting from the number one. The very word *first* is abbreviated as 1st, which indicates one. This, however, is a very modern notation. The older concepts do not necessarily support this relationship.

In English and French the word *first* is not derived from the word *one*, but from an old word for *prince*, which means *foremost*. Similarly, The English word *second* is not derived from the number two but from an old word which means "to follow." Obviously, there is a close relation between third and three, fourth and four, and so on.

These relationships occur in other language families, also. In Hebrew, for example, first is derived from the word head, meaning "the foremost." The Hebrew word for second is derived from the word two; this relationship of ordinal and cardinal names holds for all the other numbers.

For a very long time, people have counted from one, not from zero. As a matter of fact, the inclusion of zero as a full-fledged member of the set of all numbers is a relatively modern concept, even though it is one of the most important numbers mathematically. It has many important properties, such as being a multiple of any integer.

A nice mathematical theorem states that for any basis b the first b^N positive integers are represented by exactly N digits (leading zeros included). This is true if and only if the count starts with zero (hence, 0 through $b^N - 1$), not with one (for 1 through b^N).

This theorem is the basis of computer memory addressing. Typically, 2^N cells are addressed by an N-bit addressing scheme. A count starting from one rather than zero would cause the loss of either one memory cell or an additional address line. Since either price is too expensive, computer engineers agree to use the mathematical notation that starts with zero. Good for them!

This is probably the reason why all memories start at address-0, even those of systems that count bits from B1 up.

The designers of the 1401 were probably ashamed to have address-0. They hid it from the users and pretended that the memory starts at address-1.

Communication engineers, like most people, start counting from one. They never have to suffer the loss of a memory cell, for example. Therefore, they happily count one-to-eight, not zero-to-seven, as computer people do.

such that when the word W(n) is shifted right, its LSB moves into the MSB of word W($n+1$). English text strings are stored in the same order, with the first character in C0 of W0, the next in C1 of W0, and so on. This order is consistent with itself and with the English language.

The Little Endians have another scheme, which is also self-consistent. They believe in starting with the narrow end of every word and that low addresses are of lower order than high addresses. Therefore, they put their words on paper as if they were written in Hebrew:

```
...|---word2---|---word1---|---word0---|
```

When they add the bit order and the byte order, they get

```
...|---word2---|---word1---|---word0---|
....|C3,C2,C1,C0|C3,C2,C1,C0|C3,C2,C1,C0|
.....|B31.....B0|B31.....B0|B31.....B0|
```

In this regime, when word W(n) is shifted right, its LSB moves into the MSB of word W($n-1$).

English text strings are stored in the same order, with the first character in C0 of W0, the next in C1 of W0, and so on.

This order is consistent with itself, with the Hebrew language, and (more importantly) with mathematics, because significance increases with increasing item numbers (address).

Its disadvantage is that English character streams appear to be written backwards. Although this is only an aesthetic problem, it looks strange—especially to speakers of English. Little Endians respond to this by pretending to be Chinese. Instead of right-to-left, they write the bytes top-to-bottom:

```
C0: "J"
C1: "O"
C2: "H"
C3: "N"
..etc..
```

There is no significance to the notion of left and right in the bit order of a computer memory. It could as easily be up and down or mirrored by the systematic interchange of all the lefts and rights.

This idea stems not from a language model but from the concept that computer words represent numbers. By mathematical tradition, the wide end of a number (the MSB) is called left and the narrow end right. This mathematical convention is the point of reference for the designation of left and right in this discussion.

It is easy to determine whether a given computer system was designed by Little Endians or Big Endians. The keys are the way the registers are connected for the combined shift operation and for multiple precision arithmetic like integer products, how these quantities are stored in memory, and (obviously) by the order in which bytes are stored within words.

A possible point of confusion lies in the B0-to-B31 direction. Many computers were designed by Big Endians who pretended to be Little Endians rather than risk exile to Blefuscu. They used, for camouflage, the B0-to-B31 convention of the Little Endians, but kept their own conventions for bytes and words.

The PDP-10 and the IBM 360, for example, were designed by proud Big Endians: their bit, byte, word, and page orders are the same. The same order also applies to long (multiword) character strings and to multiple precision numbers.

Consider the Motorola M68000 microprocessor. It stores a 32-bit number *xy*, a 16-bit number *z*, and the string JOHN in its 16-bit words as shown below (S = sign bit, M = MSB, L = LSB):

```
 SMxxxxxxx yyyyyyyyL SMzzzzzzL  "J" "O"   "H" "N"
|--word0--|--word1--|--word2--|--word3--|--word4--|....
|-C0-|-C1-|-C0-|-C1-|-C0-|-C1-|-C0-|-C1-|-C0-|-C1-|.....
|B15....B0|B15....B0|B15....B0|B15....B0|B15....B0|......
```

The M68000 always has the wide end of numbers on the left (i.e., *lower* byte or word address) in any of the various sizes it can use: four (BCD), eight, 16, or 32 bits.

Hence, the M68000 is a consistent Big Endian, except for its bit designation, which camouflages its true identify. Remember, the Big Endians were the outlaws.

Consider the PDP-11 order, since this is the first computer claimed to be a Little Endian. Let's again look at the way data is stored in memory:

```
     "N" "H"   "O" "J"  SMzzzzzzL SMyyyyyyL SMxxxxxxL
....|--word4--|--word3--|--word2--|--word1--|--word0--|
.....|-C1-|-C0-|-C1-|-C0-|-C1-|-C0-|-C1-|-C0-|-C1-|-C0-|
......|B15....B0|B15....B0|B15....B0|B15....B0|B15....B0|
```

The PDP-11 has no instruction to move 32-bit numbers. Its multiplication products are 32-bit quantities created only in the registers. They can be stored in memory in any way. Therefore, the 32-bit quantity *xy* did not appear in the above diagram. Hence, the above is a consis-

Order of numbers

In English, we write numbers in Big Endians' left-to-right order. This is because we say numbers in the Big Endians' order and because we write English left-to-right. Mathematically, there is a lot to be said for the Little Endians' order. While serial comparators and dividers prefer the former, serial adders and multipliers prefer the latter.

When was the common Big Endian order adopted by most modern languages?

In the Bible, numbers are described in words (seven) not by digits (7), which came on the scene nearly a thousand years after the Bible was written. In the old Hebrew Bible, some numbers are expressed in Little Endian (seven and twenty and hundred), but many are in Big Endian.

When the Bible is translated into English, the contemporary English order is used. For example, "seven and twenty and hundred" appears in the Hebrew source of the Book of Esther (1:1). In the King James Version (in English) it is "hundred and seven and twenty." In the modern Revised American Standard Version it is simply "one hundred and twenty-seven."

tent Little Endian order. The PDP – 11 always stores on the left (i.e., *higher* bit or byte address) the wide end of numbers of the sizes it can use: eight or 16 bits.

Infiltration by Big Endians caused the registers of this Little Endian's marvel to be treated in the Big Endians' way: a double-length (32-bit) operand is placed with its MSB in the lower address register and the LSB in the higher address register. This means the registers must be written on paper from left to right, with the wide end of numbers in the lower address register. This affects the integer multiplication and division, the combined shifts, and more. Blefuscu scores on this one.

Later, floating-point hardware was introduced for the PDP-11/45. Floating-point numbers are represented by either 32- or 64-bit quantities, which are two or four PDP – 11 words. The wide end is the one with the sign bit(s), the exponent, and the MSB of the fraction. The narrow end is the one with the LSB of the fraction. On paper, these formats are clearly shown with the wide end on the left and the narrow end on the right, in line with centuries-old mathematical conventions. On page 12-3 of the PDP-11/45 processor handbook,[2] there is a cute graphical demonstration of this order, with the word *fraction* split over the two or the four words used to store it. But somehow the Big Endians took over, again. Blefuscuian infiltrators assigned, as they always do, the wide end to the *lower* addresses in memory and the narrow to the *higher* addresses.

Let *xy* and *abcd* be 32- and 64-bit floating-point numbers, respectively. Here is how these numbers are stored in memory:

```
ddddddddL cccccccc bbbbbbbb SMaaaaaaa yyyyyyyyL SMxxxxxxx
.|--word5--|--word4--|--word3--|--word2--|--word1--|--word0--|
..|-C1-|-C0-|-C1-|-C0-|-C1-|-C0-|-C1-|-C0-|-C1-|-C0-|-C1-|-C0-|
...|B15....B0|B15....B0|B15....B0|B15....B0|B15....B0|B15....B0|
```

The Big Endians score heavily here. The handbook[2] does not even try to camouflage it by Chinese notation.

Encouraged by this success—as minor as it is—the Blefuscuians tried to pull another fast one. This time it was on the VAX, the sacred machine worshipped by all Lilliputians. The VAX order is shown below. The same data as above (with *xy* being a 32-bit integer) is stored in memory thus:

```
      "N" "H"   "O" "J"  SMzzzzzzL SMxxxxxxx yyyyyyyyL
...ng2-------|-------long1-------|-------long0-------|
....|--word4--|--word3--|--word2--|--word1--|--word0--|
.....|-C1-|-C0-|-C1-|-C0-|-C1-|-C0-|-C1-|-C0-|-C1-|-C0-|
......|B15....B0|B15....B0|B15....B0|B15....B0|B15....B0|
```

This is a beautifully consistent Little Endian order. So what about the infiltrators? Did they fail to carry out their mission? Since the integer arithmetic was closely guarded, they attacked easier prey, the floating point and the double floating point.

Let's look, again, at the way the above data is stored, but with the 32-bit quantity *xy* changed to a floating-point number. This data is organized in memory by the customs of Blefuscu:

```
      "N" "H"   "O" "J"  SMzzzzzzL yyyyyyyyL SMxxxxxxx
...ng2-------|-------long1-------|-------long0-------|
....|--word4--|--word3--|--word2--|--word1--|--word0--|
.....|-C1-|-C0-|-C1-|-C0-|-C1-|-C0-|-C1-|-C0-|-C1-|-C0-|
......|B15....B0|B15....B0|B15....B0|B15....B0|B15....B0|
```

The VAX is found guilty, but its guilt is tempered by an extenuating circumstance: the need for compatibility with the PDP-11. The VAXians found a way around this unaesthetic appearance by using the Chinese top-to-bottom notation rather than an embarrassing left-to-right or right-to-left arrangement. Page 10 of the *VAX-11 Architecture Handbook*[3] is a marvel. One has to admire the skillful way in which some quantities are shown in columns eight bits wide, some in columns 16 bits wide, and others in columns 32 bits wide.

Some engineers complain about the vertical notation because the top of the diagrams usually corresponds to "low" memory (low addresses). However, anyone brought up by computer scientists rather than by botanists knows that trees grow downward, with roots at the top of the page and the leaves below. Computer scientists seldom remember which way up really is (see Knuth,[4] pp. 305-309).

Having scored so easily in the floating-point department, the Blefuscuians moved to packed decimal territory. The VAX is also capable of using four-bit-chunk decimal arithmetic, which is similar to the well known BCD format. The Big Endians struck again and encountered no resistance. The decimal number 12345678 is stored in the VAX memory in the following order, whose

Integers vs. fractions

Computer designers treat fixed-point multiplication in one of two ways: as an integer multiplication or as a fractional multiplication. This is because when two 16-bit numbers, for example, are multiplied, the result is a 31-bit number in a 32-bit field. Integers are right justified; fractions are left justified. The entire difference is only a single, one-bit shift. As small as this difference is, it is important. Hence, computers are wired differently for these kinds of multiplications. The addition/subtraction operation is the same for either integer or fraction operation.

If the LSB is B0, then the value of a number is

$$V = \sum_{i=0}^{15} 2^{i} B(i)$$

in the above example, which is obviously an integer. If the MSB is B0, then the value of a number is

$$V = \sum_{i=0}^{15} 2^{-i} B(i)$$

which is obviously a fraction.

After multiplication, Integerites would typically keep B0-B15 (the LSH, least significant half). They would discard the MSH after verifying that there is no overflow into it. Fractionites would also keep B0-B15, which is the MSH, but discard the LSH.

One could expect Integerites to be Little Endians and Fractionites to be Big Endians, but the world is not that consistent.

ugliness cannot be hidden even by the standard Chinese trick.

```
        7 8  5 6  3 4  1 2
...|-------long0-------|
....|--word1--|--word0--|
.....|-C1-|-C0-|-C1-|-C0-|
......|B15....B0|B15....B0|
```

What is the sum of this memory order discussion? Only the Big Endians of Blefuscu have built systems with a consistent order that works across chunk boundaries, registers, instructions, and memories. I found no totally consistent Little Endian system.

Transmission order

In either of the consistent orders the first bit (B0) of the first byte (C0) of the first word (W0) is sent first, followed by the rest of the bits of this byte, then by the rest of the bytes of this word, and so on in the same order. Such a sequence of eight 32-bit words, for example, can be viewed as either four long words, eight words, 32 bytes, or 256 bits.

For example, some people treat the Arpa-Internet datagrams as sequences of 16-bit words, while others treat them as either eight-bit byte streams or sequences of 32-bit words. This has never been a source of confusion because of the Big Endians' consistent order.

There are many ways to devise inconsistent orders. The two most popular are the following and its mirror image. Under this order, the first bit to be sent is the *least* significant bit (B0) of the *most* significant byte (C0) of the first word, followed by the rest of the bits of this byte, then by the same right-to-left bit order inside the left-to-right byte order. Figure 1 shows the transmission order for the four orders discussed above, two of which are consistent and two of which are inconsistent.

Those who use this or any other inconsistent order exclusively must consider the famous byte-order problem. If they can pretend that their communication medium is really a byte-oriented link, this inconsistency can be safely hidden under the rug.

A few years ago, eight-bit microprocessors appeared and drastically changed the way we do business. More recently, a wide variety of eight-bit communication hardware (e.g., Z80-SIO and 2652) followed. All of it operates in the Little Endians' order. Now that a wave of 16-bit microprocessors has arrived, it is conceivable that 16-bit communication hardware will soon become a reality. Since the 16-bit communication gear will be provided by the same folks who brought us the eight-bit gear, it is safe to expect these two modes to be compatible.

The only way to achieve compatibility is by using a consistent Little Endian order, since all the existing gear is already in Little Endian. We have observed that the Little Endians do not have consistent memory orders for intracomputer organization.

If the 16-bit communication link can be made to operate in any order, consistent or not, which would give it the appearance of being a byte-oriented link, *then* the Big Endians could push (ask? hope? pray?) for an order that transmits the bytes left-to-right (i.e., wide end first) and use that as a basis for transmitting all quantities except BCD in the more convenient Big Endian format. This format has several advantages, including the fact that the most significant portions would lead the least significant. Furthermore, compatibility would be maintained between 16- and 32-bit communication. The *if*, however, is a big one.

Wouldn't it be nice if we could encapsulate the byte communication and forget about the established idiosyncrasy (introduced by RS-232 and Telex) of sending the narrow end first? It would be nice, but nice things do not necessarily occur, especially with so much silicon stacked against them.

Hence, our choice now is between (1) Big Endian's computer-convenience and (2) future compatibility among communication devices handling different chunk sizes. Short-term convenience considerations favor the former; long-term considerations favor the latter.

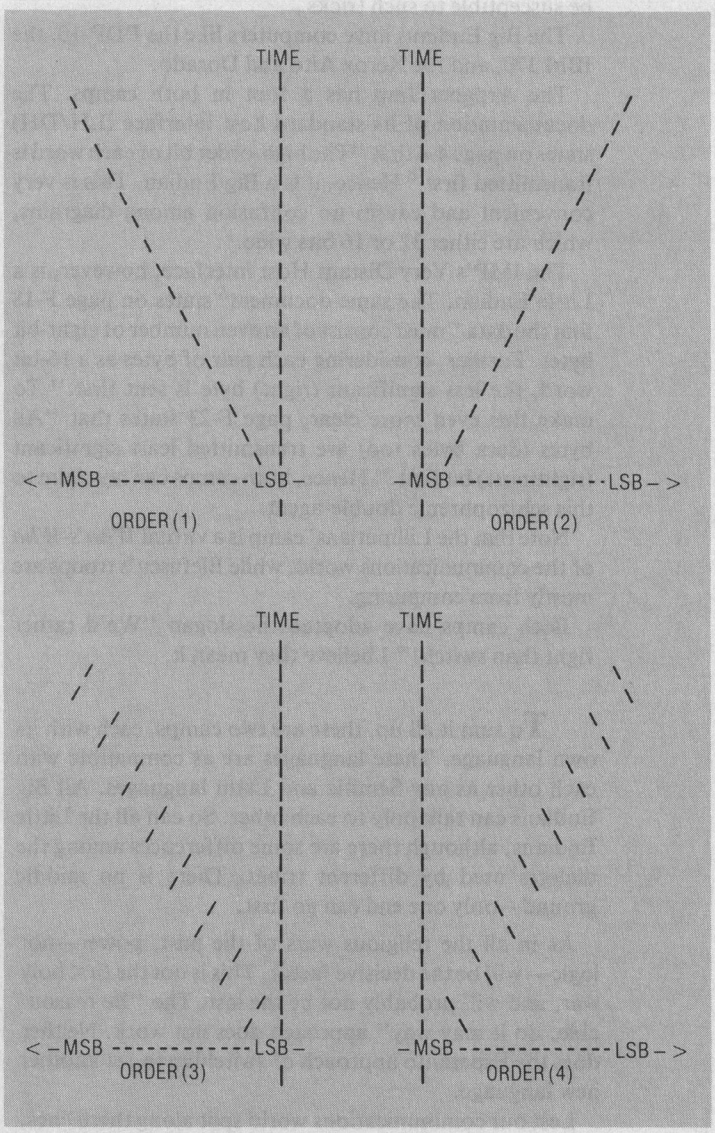


Figure 1. Possible orders. Consistent: (1) + (2); inconsistent: (3) + (4).

The lines of battle

Since war between the Little Endians and the Big Endians is imminent, let's count the troops in each camp. The founders of the Little Endians are RS-232 and Telex, which send the narrow end first. So do the HDLC and the SDLC protocols, the Z80-SIO, the Signetics-2652, the Intel 8251, the Motorola 6850, and all the rest of the existing communication devices. In addition to these protocols and chips, the PDP-11s and the VAXes have already pledged allegiance to this camp.

The HDLC protocol is a full-fledged Little Endian because it sends all of its fields with the narrow end first, as specifically defined in Table 1/X.25 (frame formats) in section 2.2.1 of *Recommendation X.25*.[5] A close examination of this table reveals that the bit order of transmission is always one-to-eight, except in the FCS (checksum) field, which is only 16-bit quantity in the byte-oriented protocol.

The FCS is sent in the 16-to-1 order. No wonder the Blefuscuians succeeded in such a coup. Anyone who designates bits as one-to-eight instead of zero-to-seven must be susceptible to such tricks.

The Big Endians have computers like the PDP-10, the IBM 370, and the Xerox Alto and Dorado.

The Arpanet Imp has a foot in both camps. The documentation of its standard host interface (LH/DH) states on page 4-4 that "The high-order bit of each word is transmitted first.[6] Hence, it is a Big Endian. This is very convenient and causes no confusion among diagrams, which are either 32 or 16 bits wide.

The IMP's Very Distant Host interface, however, is a Little Endian. The same document[6] states on page F-18 that the data "must consist of an even number of eight-bit bytes. Further, considering each pair of bytes as a 16-bit word, the less significant (right) byte is sent first." To make this even more clear, page F-23 states that "All bytes (data bytes too) are transmitted least significant (rightmost) bit first." Hence, both camps can lay claim to this schizophrenic double-agent.

Note that the Lilliputians' camp is a virtual *Who's-Who* of the communications world, while Blefuscu's troops are mostly from computing.

Both camps have adopted the slogan "We'd rather fight than switch!" I believe they mean it.

To sum it all up, there are two camps, each with its own language. These languages are as compatible with each other as any Semitic and Latin languages. All Big Endians can talk only to each other. So can all the Little Endians, although there are some differences among the dialects used by different tribes. There is no middle ground—only one end can go first.

As in all the religious wars of the past, power—not logic— will be the decisive factor. This is not the first holy war, and will probably not be the last. The "Be reasonable, do it may way" approach does not work. Neither does the Esperanto approach of switching to yet another new language.

Lest our communications world split along these lines, we should take note of a certain book (not mentioned in the references), which has an interesting story about a similar phenomenon: the Tower of Babel.

Lilliput and Blefuscu will never come to terms of their own free will. We need some Gulliver between the two islands to force a unified communication regime on all of us. Of course, I hope that my way will be chosen, but it is not really critical. Agreement upon an order is more important than the order agreed upon.

Shall we toss a coin?■

References

1. Jonathan Swift, *Gulliver's Travels* (original title: *Travels into several Remote Nations of the World. By Lemuel Gulliver*), London, Benjamin Mott, 1726.
2. *PDP-11 04/05/10/35/40/45 Processor Handbook*, Digital Equipment Corp., Maynard, Mass., 1975.
3. *VAX-11—Architecture Handbook*, Digital Equipment Corp., Maynard, Mass., 1979.
4. D. E. Knuth, *The Art of Computer Programming*, Vol. I, Addison-Wesley, Reading, Mass., 1968.
5. *CCITT Orange Book, Vol. VIII.2: Public Data Networks*, International Telecommunication Union, Geneva, 1977.
6. *Interface Message Processor*, Report No. 1822, Bolt Beranek and Newman, Cambridge, Mass., May 1978.

Danny Cohen is a project leader at the Information Sciences Institute of the University of Southern California, where he has worked since 1973. His research interests include interactive real-time systems, graphics, voice communications, networking, VLSI, flight simulation, and computer architecture. Before joining ISI, he served on the faculties of Harvard University, the Technion (Israel Institute of Technology), and the California Institute of Technology.

Cohen received the BSc degree in mathematics from the Technion-Israel Institute of Technology in 1963 and the PhD degree in computer science from Harvard University in 1969.