IEN 137

Danny Cohen
U S C/I S I
1 April 1980

ON HOLY WARS AND A PLEA FOR PEACE

INTRODUCTION

This is an attempt to stop a war. I hope it is not too late and that somehow, magically perhaps, peace will prevail again.

The latecomers into the arena believe that the issue is: "What is the proper byte order in messages?".

The root of the conflict lies much deeper than that. It is the question of which bit should travel first, the bit from the little end of the word, or the bit from the big end of the word? The followers of the former approach are called the Little-Endians, and the followers of the latter are called the Big-Endians. The details of the holy war between the Little-Endians and the Big-Endians are documented in [6] and described, in brief, in the Appendix. I recommend that you read it at this point.

The above question arises from the serialization process which is performed on messages in order to send them through communication media. If the communication unit is a message - these problems have no meaning. If the units are computer "words" then one may ask in which order these words are sent, what is their size, but not in which order the elements of these words are sent, since they are sent virtually "at-once". If the unit of transmission is an 8-bit byte, similar questions about bytes are meaningful, but not the order of the elementary particles which constitute these bytes.

If the units of communication are bits, the "atoms" ("quarks"?) of computation, then the only meaningful question is the order in which bits are sent.

Obviously, this is actually the case for serial transmission. Most modern communication is based on a single stream of information ("bit-stream"). Hence, bits, rather than bytes or words, are the units of information which are actually transmitted over the communication channels such as wires and satellite connections.

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

Computer memory may be viewed as a linear sequence of bits, divided into bytes, words, pages and so on. Each unit is a subunit of the next level. This is, obviously, a hierarchical organization.

If the order is consistent, then such a sequence may be communicated successfully while both parties maintain their freedom to treat the bits as a set of groups of any arbitrary size. One party may treat a message as a "page", another as so many "words", or so many "bytes" or so many

bits. If a consistent bit order is used, the "chunk-size" is of no consequence.

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

In a consistent order, the bit-order, the byte-order, the word-order, the page-order, and all the other higher level orders are all the same. Hence, when considering a serial bit-stream, along a communication line for example, the "chunk" size which the originator of that stream has in mind is not important.

There are two possible consistent orders. One is starting with the narrow end of each word (aka "LSB") as the Little-Endians do, or starting with the wide end (aka "MSB") as their rivals, the Big-Endians, do.

In this note we usually use the following sample numbers: a "word" is a 32-bit quantity and is designated by a "W", and a "byte" is an 8-bit quantity which is designated by a "C" (for "Character", not to be confused with "B" for "Bit)".

MEMORY ORDER

The first word in memory is designated as W0, by both regimes. Unfortunately, the harmony goes no further.

The Little-Endians assign B0 to the LSB of the words and B31 is 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 up, not from ONE, as is traditionally done. If so, 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 either the Little-Endians' or the Big-Endians' order. These people probably would like to number their words from W1 up, just to be consistent.

Back to the main theme. We would like to illustrate the hierarchically consistent order graphically, but first we have to decide about the order in which computer words are written on paper. Do they go from left to right, or from right to left?

The English language, like most modern languages, suggests that we lay these computer words on paper from left to right, like this:

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

In order 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 is also to the left of C3. Hence we get:

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

```
|B0......B31|B0......B31|B0......B31|......
```

If we also use the traditional convention, as introduced by our numbering system, the wide-end is on the left and the narrow-end is on the right.

Hence, the above is a perfectly consistent view of the world as depicted by the Big-Endians. Significance consistency decreases as the item numbers (address) increases.

Many computers share with the Big-Endians this view about order. In many of their diagrams the registers are connected 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 very consistent with itself and with the English language.

On the other hand, the Little-Endians have their view, which is different but also self-consistent.

They believe that one should start 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, like this:

```
...|---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 very consistent with itself, with the Hebrew language, and (more importantly) with mathematics, because significance increases with increasing item numbers (address).

It has the disadvantage that English character streams appear to be written backwards; this is only an aesthetic problem but, admittedly, it looks funny, especially to speakers of English.

In order to avoid receiving strange comments about this orders the Little-Endians pretend that they are Chinese, and write the bytes, not right-to-left but top-to-bottom, like:

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

Note that there is absolutely no specific significance whatsoever to the notion of "left" and "right" in bit order in a computer memory. One could think about it as "up" and "down" for example, or mirror it by systematically interchanging all the "left"s and "right"s. However, this notion stems from the concept that computer words represent numbers, and from the old mathematical tradition that the wide-end of a number (aka the MSB) is called "left" and the narrow-end of a number is called "right".

This mathematical convention is the point of reference for the notion of "left" and "right".

It is easy to determine whether any given computer system was designed by Little-Endians or by Big-Endians. This is done by watching the way the registers are connected for the "COMBINED-SHIFT" operation and for multiple-precision arithmetic like integer products; also by watching how these quantities are stored in memory; and obviously also by the order in which bytes are stored within words. Don't let the B0-to-B31 direction fool you!! Most computers were designed by Big-Endians, who under the threat of criminal prosecution pretended to be Little-Endians, rather than seeking exile in Blefuscu. They did it by using the B0-to-B31 convention of the Little-Endians, while keeping the Big-Endians' conventions for bytes and words.

The PDP10 and the 360, for example, were designed by Big-Endians: their bit order, byte-order, word-order and page-order are the same. The same order also applies to long (multi-word) character strings and to multiple precision numbers.

Next, let's consider the new M68000 microprocessor. Its way of storing a 32-bit number, xy, a 16-bit number, z, and the string "JOHN" in its 16-bit words is 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 on the left (i.e., LOWER byte- or word-address) the wide-end of numbers in any of the various sizes which it may use: 4 (BCD), 8, 16 or 32 bits.

Hence, the M68000 is a consistent Big-Endian, except for its bit designation, which is used to camouflage its true identity. Remember: the Big-Endians were the outlaws.

Let's look next at the PDP11 order, since this is the first computer to claim 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 PDP11 does not have an instruction to move 32-bit numbers. Its multiplication products are 32-bit quantities created only in the registers, and may be stored in memory in any way. Therefore, the

32-bit quantity, xy, was not shown in the above diagram.

Hence, the above order is a Little-Endians' consistent order. The PDP11 always stores on the left (i.e., HIGHER bit- or byte-address) the wide-end of numbers of any of the sizes which it may use: 8 or 16 bits.

However, due to some infiltration from the other camp, the registers of this Little-Endian's marvel are treated in the Big-Endians' way: a double length operand (32-bit) is placed with its MSB in the lower address register and the LSB in the higher address register. Hence, when depicted on paper, the registers have to be put 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. Admittedly, Blefuscu scores on this one.

Later, floating-point hardware was introduced for the PDP11/45.

Floating-point numbers are represented by either 32- or 64-bit quantities, which are 2 or 4 PDP11 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 on the right, according to the centuries old mathematical conventions. On page 12-3

of the PDP11/45 processor handbook, [3], there is a cute graphical demonstration of this order, with the word "FRACTION" split over all the 2 or the 4 words which are used to store it.

However, due to some oversights in the security screening process, the Blefuscuians took over, again. They 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. Let's look how these numbers are stored in memory:

```
          ddddddddL ccccccccc bbbbbbbbb 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|
```

Well, Blefuscu scores many points for this. The above reference in [3] does not even try to camouflage it by any 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 which all the Little-Endians worship.

Let's look at the VAX order. Again, we look at the way the above data (with xy being a 32-bit integer) is stored in memory:

```
              "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|
```

What a beautifully consistent Little-Endians' order this is !!!

So, what about the infiltrators? Did they completely fail in carrying out their mission? Since the integer arithmetic was closely guarded they attacked the floating point and the double-floating which were already known to be easy prey.

Let's look, again, at the way the above data is stored, except that now the 32-bit quantity xy is a floating point number: now this data is organized in memory in the following Blefuscuian way:

```
              "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|
```

Blefuscu scores again. The VAX is found guilty, however with the explanation that it tries to be compatible with the PDP11.

Having found themselves there, the VAXians found a way around this unaesthetic appearance: the VAX literature (e.g., p. 10 of [4]) describes this order by using the Chinese top-to-bottom notation, rather than an embarrassing left-to-right or right-to-left one. This page is a marvel. One has to admire the skillful way in which some quantities are shown in columns 8-bit wide, some in 16 and other in 32, all in order to avoid the egg-on-the-face problem.....

By the way, some engineering-type people complain about the "Chinese" (vertical) notation because usually the top (aka "up") of the diagrams corresponds to "low"-memory (low addresses). However, anyone who was brought up by computer scientists, rather than by botanists, knows that trees grow downward, having their roots at the top of the page and their leaves down below. Computer scientists seldom remember which way "up" really is (see 2.3 of [5], pp. 305-309).

Having scored so easily in the floating point department, the Blefuscuians moved to new territories: Packed-Decimal. The VAX is also capable of using 4-bit-chunk decimal arithmetic, which is similar to the well known BCD format.

The Big-Endians struck again, and without any resistance got their way. The decimal number 12345678 is stored in the VAX memory in this order:

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

This ugliness cannot be hidden even by the standard Chinese trick.

SUMMARY (of the Memory-Order section)

To the best of my knowledge only the Big-Endians of Blefuscu have built

systems with a consistent order which works across chunk-boundaries, registers, instructions and memories. I failed to find a Little-Endians' system which is totally consistent.

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, then the rest of the bits of this byte, then (in the same order) the rest of the bytes of this word, and so on.

Such a sequence of 8 32-bit words, for example, may be viewed as either 4 long-words, 8 words, 32 bytes or 256 bits.

For example, some people treat the ARPA-internet-datagrams as a sequence of 16-bit words whereas others treat them as either 8-bit byte streams or sequences of 32-bit words. This has never been a source of confusion, because the Big-Endians' consistent order has been assumed.

There are many ways to devise inconsistent orders. The two most popular ones 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 the same right-to-left bit order inside the left-to-right byte order.

Figure 1 shows the transmission order for the 4 orders which were discussed above, the 2 consistent and the 2 inconsistent ones.

Those who use such an inconsistent order (or any other), and only those, have to be concerned with the famous byte-order problem. If they can pretend that their communication medium is really a byte-oriented link then this inconsistency can be safely hidden under the rug.

A few years ago 8-bit microprocessors appeared and changed drastically the way we do business. A few years later a wide variety of 8-bit communication hardware (e.g., Z80-SIO and 2652) followed, all of which operate in the Little-Endians' order.

Now a wave of 16-bit microprocessors has arrived. It is not inconceivable that 16-bit communication hardware will become a reality relatively soon.

Since the 16-bit communication gear will be provided by the same folks who brought us the 8-bit communication gear, it is safe to expect these two modes to be compatible with each other.

The only way to achieve this is by using the consistent Little-Endians order, since all the existing gear is already in Little-Endians order.

We have already observed that the Little-Endians do not have consistent memory orders for intra-computer organization.

IF the 16-bit communication link could 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 which transmits the bytes in 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-Endians format, with the most significant portions leading the least significant, maintaining compatibility between 16- and 32-bit communication, and more.

However, this is a big "IF".

Wouldn't it be nice if we could encapsulate the byte-communication and forget all about the idiosyncrasies of the past, introduced by RS232 and TELEX, of sending the narrow-end first?

I believe that it would be nice, but nice things do not necessarily occur, especially if there is so much silicon against them.

Hence, our choice now is between (1) Big-Endians' computer-convenience and (2) future compatibility between communication gear of different chunk size.

I believe that this is the question, and we should address it as such.

Short term convenience considerations are in favor of the former, and the long term ones are in favor of the latter.

Since the war between the Little-Endians and the Big-Endians is imminent, let's count who is in whose camp.

The founders of the Little-Endians party are RS232 and TELEX, who stated that the narrow-end is sent first. So do the HDLC and the SDLC protocols, the Z80-SIO, Signetics-2652, Intel-8251, Motorola-6850 and all the rest of the existing communication devices. In addition to these protocols and chips the PDP11s and the VAXes have already pledged their allegiance to this camp, and deserve to be on this roster.

1

The HDLC protocol is a full fledged member of this camp because it sends all of its fields with the narrow end first, as is specifically defined in Table 1/X.25 (Frame formats) in section 2.2.1 of Recommendation X.25 (see [2]). A close examination of this table reveals that the bit order of transmission is always 1-to-8. Always, except the FCS (checksum) field, which is the only 16-bit quantity in the byte-oriented protocol.

The FCS is sent in the 16-to-1 order. How did the Blefuscuians manage to pull off such a fiasco?! The answer is beyond me. Anyway, anyone who designates bits as 1-to-8 (instead of 0-to-7) must be gullible to such tricks.

The Big-Endians have the PDP10's, 370's, ALTO's and Dorado's...

An interesting creature is the ARPANet-IMP. The documentation of its standard host interface (aka "LH/DH") states that "The high order bit of each word is transmitted first" (p. 4-4 of [1]), hence, it is a Big-Endian. This is very convenient, and causes no confusion between diagrams which are either 32- (e.g., on p. 3-25) and 16-bit wide (e.g., on p. 5-14).

However, the IMP's Very Distant Host (VDH) interface is a Little-Endian.

The same document ([1], again, p. F-18), states that the data "must

consist of an even number of 8-bit bytes. Further, considering each pair of bytes as a 16-bit word, the less significant (right) byte is sent first".

In order to make this even more clear, p. F-23 states "All bytes (data bytes too) are transmitted least significant (rightmost) bit first".

Hence, both camps may claim to have this schizophrenic double-agent in their camp.

Note that the Lilliputians' camp includes all the who's-who of the communication world, unlike the Blefuscuians' camp which is very much oriented toward the computing world.

Both camps have already adopted the slogan "We'd rather fight than switch!".

I believe they mean it.

SUMMARY (of the Transmission-Order section)

There are two camps each with its own language. These languages are as compatible with each other as any Semitic and Latin languages are.

All Big-Endians can talk to each other with relative ease.

So can all the Little-Endians, even though there are some differences among the dialects used by different tribes.

There is no middle ground. Only one end can go first.

CONCLUSION

Each camp tries to convert the other. Like all the religious wars of the past, logic is not the decisive tool. Power is. This holy war is not the first one, and probably will not be the last one either.

The "Be reasonable, do it my way" approach does not work. Neither does the Esperanto approach of "let's all switch to yet a new language".

Our communication world may split according to the language used. A certain book (which is NOT mentioned in the references list) has an interesting story about a similar phenomenon, the Tower of Babel.

Little-Endians are Little-Endians and Big-Endians are Big-Endians and never the twain shall meet.

We would like to see some Gulliver standing up between the two islands, forcing a unified communication regime on all of us. I do hope that my way will be chosen, but I believe that, after all, which way is chosen

does not make too much difference. It is more important to agree upon an order than which order is agreed upon.

How about tossing a coin ???

1

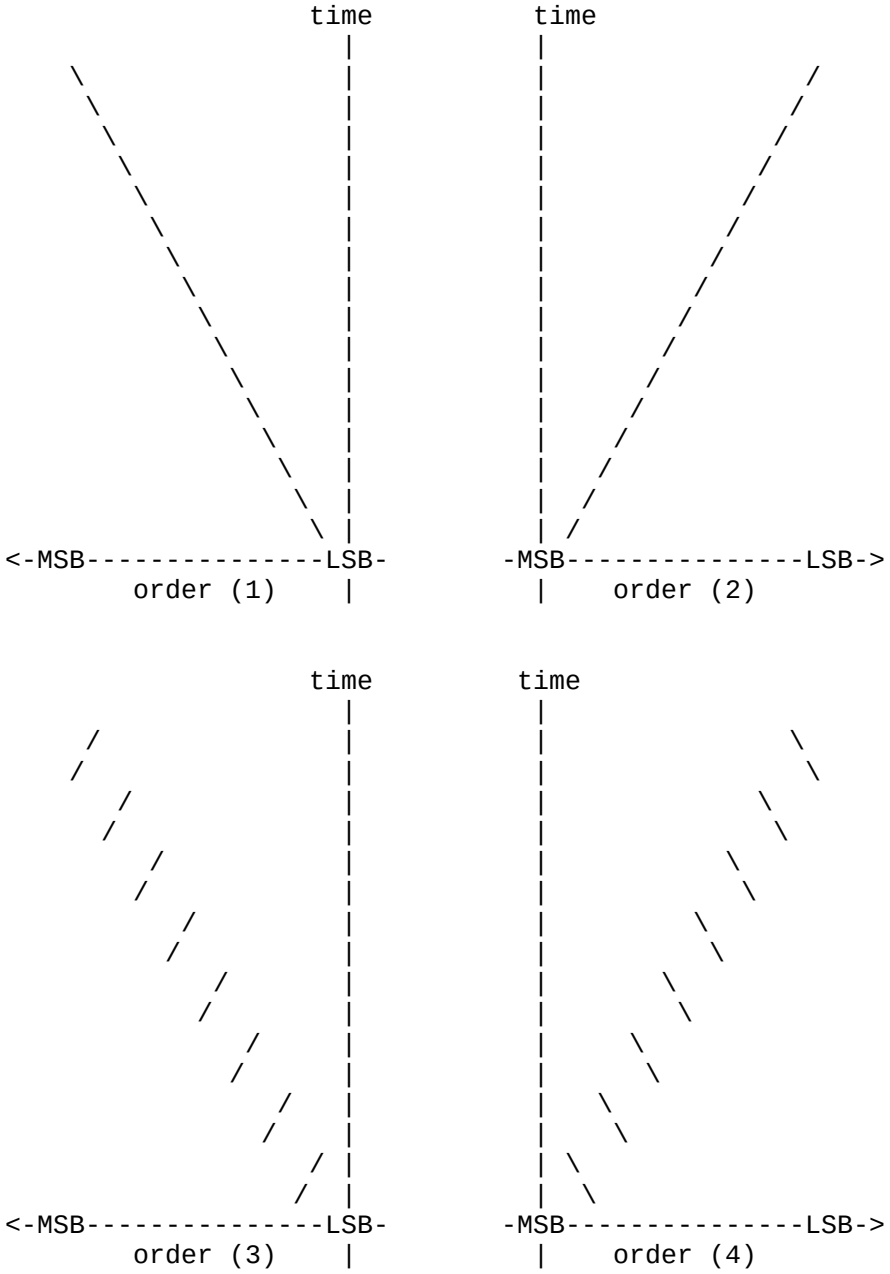

Figure 1: Possible orders, consistent: (1)+(2), inconsistent: (3)+(4).

1

A P P E N D I X

Some notes on Swift's Gulliver's Travels:

Gulliver finds out that there is a law, proclaimed by the grandfather of the present ruler, requiring all citizens of Lilliput to break their eggs only at the little ends. Of course, all those citizens who broke their eggs at the big ends were angered by the proclamation. Civil war broke out between the Little-Endians and the Big-Endians, resulting in the Big-Endians taking refuge on a nearby island, the kingdom of Blefuscu.

Using Gulliver's unquestioning point of view, Swift satirizes religious wars. For 11,000 Lilliputian rebels to die over a controversy as trivial as at which end eggs have to be broken seems not only cruel but also absurd, since Gulliver is sufficiently gullible to believe in the significance of the egg question. The controversy is important ethically and politically for the Lilliputians. The reader may think the issue is silly, but he should consider what Swift is making fun of the actual causes of religious- or holy-wars.

In political terms, Lilliput represents England and Blefuscu France. The religious controversy over egg-breaking parallels the struggle between the Protestant Church of England and the Catholic Church of France, possibly referring to some differences about what the Sacraments really mean. More specifically, the quarrel about egg-breaking may allude to the different ways that the Anglican and Catholic Churches distribute communion, bread and wine for the Anglican, but bread alone for the Catholic. The French and English struggled over more mundane questions as well, but in this part of Gulliver's Travels, Swift points up the symbolic difference between the churches to ridicule any religious war.

For ease of reference please note that Lilliput and Little-Endians both start with an "L", and that both Blefuscu and Big-Endians start with a "B". This is handy while reading this note.

R E F E R E N C E S

[1] Bolt Beranek & Newman.
Report No. 1822: Interface Message Processor.
Technical Report, BB&N, May, 1978.

[2] CCITT.
Orange Book. Volume VIII.2: Public Data Networks.
International Telecommunication Union, Geneva, 1977.

[3] DEC.
PDP11 04/05/10/35/40/45 processor handbook.
Digital Equipment Corp., 1975.

[4] DEC.
VAX11 - Architecture Handbook.
Digital Equipment Corp., 1979.

[5] Knuth, D. E.

The Art of Computer Programming. Volume I: Fundamental
Algorithms.
Addison-Wesley, 1968.

[6] Swift, Jonathan.
Gulliver's Travel.
Unknown publisher, 1726.

1

OTHER SLIGHTLY RELATED TOPICS (IF AT ALL)

not necessarily for inclusion in this note

Who's on first? Zero or One ??

People start counting from the number ONE. The very word FIRST is abbreviated into the symbol "1st" which indicates ONE, but this is a very modern notation. The older notions 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 an close relation between "third" and "three", "fourth" and "four" and so on.

Similarly, in Hebrew, for example, the word "first" is derived from the word "head", meaning "the foremost", but not specifically No. 1. The Hebrew word for "second" is specifically derived from the word "two". The same for three, four and all the other numbers.

However, people have,for a very long time, counted from the number 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.

Zero 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 (b to the Nth power) 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. Starting the count from One, rather than Zero, would cause either the loss of one memory cell, or an additional address line. Since either price is too expensive, computer engineers agree to use the mathematical notation of starting with Zero. Good for them!

The designers of the 1401 were probably ashamed to have address-0 and hid it from the users, pretending that the memory started at address-1.

1

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

Communication engineers, like most "normal" people, start counting from the number One. They never suffer by having to lose a memory cell, for example. Therefore, they are happily counting 1-to-8, and not 0-to-7 as computer people learn to do.

ORDER OF NUMBERS.

In English, we write numbers in Big-Endians' left-to-right order. I believe that this is because we SAY numbers in the Big-Endians' order, and because we WRITE English in Left-to-right order.

Mathematically there is a lot to be said for the Little-Endians' order.

Serial comparators and dividers prefer the former. Serial adders and multipliers prefer the latter order.

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

In the Bible, numbers are described in words (like "seven") not by digits (like "7") which were "invented" nearly a thousand years after the Bible was written. In the old Hebrew Bible many numbers are expressed in the Little-Endians order (like "Seven and Twenty and Hundred") but many are in the Big-Endians order as well.

Whenever the Bible is translated into English the contemporary English order is used. For example, the above number appears in that order in the Hebrew source of The Book of Esther (1:1). In the King James Version it is (in English) "Hundred and Seven and Twenty". In the modern Revised American Standard Version of the Bible this number is simply "One Hundred and Twenty-Seven".

INTEGERS vs. FRACTIONS

Computer designers treat fix-point multiplication in one of two ways, as an integer-multiplication or as a fractional-multiplication.

The reason is that 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 1-bit shift. As small as it is, this is an important difference.

Hence, computers are wired differently for these kinds of multiplications. The addition/subtraction operation is the same for either integer/fraction operation.

1

If the LSB is B0 then the value of a number is SIGMA<B(i)*[(2)^i]>, for i=0,15, in the above example. This is, obviously, an integer.

If the MSB is B0 then the value of a number is SIGMA<B(i)*[(1/2)^i]>,

for i=0,15. This is, obviously, a fraction.

Hence, after multiplication the Integerites would typically keep B0-B15, the LSH (Least Significant Half), and discard the MSH, after verifying that there is no overflow into it. The Fractionites would also keep B0-B15, which is the MSH, and discard the LSH.

One could expect Integerites to be Little-Endians, and Fractionites to be Big-Endians. I do not believe that the world is that consistent.

SWIFT's POINT

It may be interesting to notice that the point which Jonathan Swift tried to convey in Gulliver's Travels in exactly the opposite of the point of this note.

Swift's point is that the difference between breaking the egg at the little-end and breaking it at the big-end is trivial. Therefore, he suggests, that everyone does it in his own preferred way.

We agree that the difference between sending eggs with the little- or the big-end first is trivial, but we insist that everyone must do it in the same way, to avoid anarchy. Since the difference is trivial we may choose either way, but a decision must be made.