BARRY BLESSER

Words

• Assuming that I have been successful in convincing you that the key to both understanding and communications is a clear sense of word meaning, I would like to present a glossary of words and definitions. If you think that these are obvious, try giving a colleague a test and compare his answers to those below.

1. BIT

The smallest unit of information in a digital system. A bit is determined as having one of two possible states to carry information. In digital hardware, a bit will be a specific wire, flip-flop, or time slot when time sharing. In communications, a bit is simply a measure of information; for example, that signal has an information rate of 20,000 bits/second. The signal may actually be an analog signal but its information can be measured in bits.

2. BYTE

A group of bits of some specific length forming a unit of information. Historically, the size needed to be specified. Present usage of byte is always referred to an 8-bit group.

3. WORD

Historically, this had been the largest group of bits to form a unit. It then came to mean a 16-bit unit because much of the hardware had 16-bit bus structures. The current usage requires a specification of the size. For example, we can say that the new microprocessors use 32-bit words for internal computation. This means that the largest unit that can be moved is 32 bits. Sometimes the size of a word is measured in terms of the number of bytes. We can say that the system has a 4-byte instruction word instead of saying it has 32 bits.

4. NIBBLE

This is a relatively new term which, like a byte, refers to a specific group of bits; however, it is used to mean 4 bits. It came from the fact that 8-bit microprocessors had instruction to manipulate the upper and lower 4 bits independently. For example, we can say the upper nibble is transferred to the I/O port.

5. INTEGER ARITHMETIC

This refers to the way in which the bits of a word can be used to represent numbers. An integer number means that the bits map directly to a sequence of discrete numbers. A 16-bit word can represent any integer in the range of +32,767 to -32,768. No number outside of that range can be represented, nor can fractions be represented. These cases do not exist in a 16-bit integer representation. Integer arithmetic can use different quantities of bits. With 8 bits, only numbers from 127 to -128 can be represented.

6. MAGNITUDE REPRESENTATION

A number system which does not contain any negative numbers is defined as a magnitude representation. With 16-bit integer magnitude representation, we can represent the numbers from 0 to 65,536. The example above (5) described numbers with both positive and negative signs. The same sequence of binary bits has a different value depending on the definition of representation. Without knowing the representation we can not know the "true" value of the number.

7. NEGATIVE NUMBERS

Negative numbers have had two different representation systems: 2's complement and 1's complement, although the latter is rarely used now. In a 2's complement number, negative numbers are created from positive numbers by complementing the positive number and adding 1 LSB (Least Significant Bit) to the result.

This has many nice properties (including the fact that the negative of 0 is itself 0 since the complement first produces all 1's and the adding of the 1 LSB results in 0 again.

A 2's complement system has only one artifact: there is one more negative number than positive number. This can be seen clearly since there is an even number of integers and one of these is used for 0. There is no positive of the most negative number.

A 1's complement system made negative numbers by the complement of positive numbers. This posed the major problem of having two numbers for 0: 000000, and 1111111. The computer hardware was thus made more complex and this system was dropped. The advantage of the system was that negation did not require an addition of the 1 LSB. It also had the advantage of having the same number of positive and negative numbers. Nevertheless, this system is rarely used in new equipment.

8. DOUBLE PRECISION INTEGER

When the range of integers represented by one word is insufficient, an additional word is used. This greatly expands the range of numbers. For example, a 32-bit 2's complement integer word (double precision) has a maximum integer of 2,147,483,648. The term double precision is actually only used to mean twice the precision of a lower precision alternative. A 16-bit representation is double precision in a system which can also use 8-bit arithmetic. In systems which have 16- and 32-bit choices, the 16-bit is single precision and the 32-bit is double precision.

Sometimes other terms are used, such as long integers and short integers. Extended precision can sometimes mean the same thing as double precision, although the increase may not be a full doubling. For example, single precision might be 3 bytes and extended precision 5 bytes.

9. FLOATING POINT

In some scientific applications, the range of numbers is much larger than that which can be found in integers. A physics problem might use measures corresponding to the size of atoms or the size of the universe. Therefore we use a scientific type notation in which we have a number scaled by some exponent. This can be written .23343 E12, which means the actual number is $.23343 \times 10^{12}$. The resolution of this number system is determined by the fractional part. There might only be 16 bits in the fraction. This means that the fractional part is quantized and only the following are possible legal fractions: 0000305, .0000610, .0000916, .0001221......... .9999390, .9999695. The exponent of the power of 10 might have a limit from +127 to -128.

Some bits in the word are removed from the fractional part to create a wider range of numbers. The smallest typical floating point number would use 8 bits for the exponent and 24 for the fractional part. Most systems use a full 6 bytes for a floating point number.

The term floating point comes from the fact that the decimal point floats and is not fixed. The exponent effectively moves the location of the decimal point. The two numbers, .23343 E2 and .23343 E3, differ by the location of the decimal point. One number is 23.343 and the other is 233.43.

In the previous discussion we implied that integer numbers have no fractional part. We may define a fixed scale factor to allow for fractions. When we do that we have created a fixed point number. With 16 bits, we can place the binary point (used to separate the integer from the fractional part, as is the decimal point) between the upper and lower bytes. This would create a range of numbers which were the following: .0, .0030, .0078......, .99609, 1.00000, 1.0030, 1.0078........, 127.9922, 127.9961.

10. DOUBLE PRECISION FLOATING POINT

An alternative version of a floating point number with additional bits in the representation to allow for better resolution in the fractional part and wider range in the exponent.

11. BINARY NUMBERS

Numbers which are *written* or *stored* as a set of binary bits. The actual *value* of the binary number is defined by the representation system. If 1,101,111,110,001 is a binary number, we cannot know the value of the number without knowing the representation such as fixed point integer, 2's complement, floating point, etc.

12. OCTAL NUMBERS

These are numbers in which groups of 3 binary bits have been merged together to make the number more complex. It allows for writing a binary number more comfortably. The example above would then become 15,761 in octal format. This is transposed as follows:

binary:	1	101	111	110	001
octal:	1	5	7	6	1

Octal numbers are really binary numbers and vice versa. Each digit in the octal number can have one of 8 values, which we name as the numbers 0 through 7. Each column of digits has 8 times the value of the previous one.

One can learn to count, add, subtract, multiply, and divide in an octal numbering system.

13. HEX NUMBERS

Hex is short for hexadecimal which means that there are 16 numbers for each digit. Like octal it is a group of bits, except that the grouping is 4 rather than 3. Each digit can have values of 0 through 15. The numbers 10, 11, 12, 13, 14, and 15 create a notational problem because they require two digits in our normal notation of 10's. To make these numbers single entries, they are given the symbols A, B, C, D, E, and F, respectively. It takes some practice to think of a number written as 1BF1, but this is the same as the previous example of 15,761 in octal. Hex notation is comfortable when there are 8 bits (two hex digits), 16 bits (4 hex digits), or any number divisible by 4.

14. ASCII

ASCII is a standard international mapping between the letters and number symbols of a 7-bit word. Each letter, number, and control symbol is given a unique value. When the key on a computer keyboard is pushed, that 7-bit word is generated. The actual word contains 8 bits; the extra bit is often used for parity checking (even or odd), or sometimes fixed as a leading 1 or 0.

ASCII is another representation system for binary numbers. This further illustrates the reason why we cannot tell the value of a binary number without knowing the system. The number could be the symbol for a letter such as Q. We say that the value or meaning is a letter rather than a quantity.

15. OVERFLOW

When the result of an arithemtic operation of two legal numbers gives a number too large to be represented, the result is called an overflow error. The term overflow is a particular kind of error which cannot be avoided. In an 8-bit integer system, there is no legal result when 127 is added to 2 because the number 129 does not exist.

Overflow can apply to both fixed and floating point numbers.

16. UNDERFLOW

This is the reverse case of overflow: the result is too small. It is generally only applied to floating point numbers which have special problems when there is not enough range in the exponent. The fractional part is limited to the range of 1 to 0.5. Therefore, the smallest number might be 0.5 E-99. An underflow would result when we tried to divide this number by 2. The result would be the same as the initial number since there is nothing smaller.

17. CLIPPING

What audio engineers call clipping, digital engineers call limiting. A hardware of software trap is added to the arithmetic process such as that the overflow error case is treated explicitly. The result is artificial, limited (clipped) to the most positive or negative value. We call this an "error handling" algorithm. The overflow is an error, but we define the way in which it will be handled. The previous example, for adding 127 to 2, gives the result 127 when we have clipping. Similarly, 120 plus 12 gives 127 and -110 plus -30 gives -128.

18. TRUNCATION

This is the name for the process by which certain bits in the result are thrown away. The resulting error is called a truncation error. It typically results from multiplication and division since these processes double the size of the result. A 16 bit number multiplied by a 16 bit number result in 31 bits. If the result must be again represented in 16 bits, we throw away the lower 15. The act of discarding means that the result contains an unknown error. It is analogous to quantization error in an A/D converter.

Addition and subtraction do not result in truncation issues because the result has the same number of bits. The exception is the overflow case. There the upper bit is lost because the number is too small. If the addition were to have a *defined* divide by 2, then there could never be an overflow, but there would be a truncation error since the lowest bit is removed. The addition of two 16-bit numbers can result in a 17-bit answer. The extra bit is usually an issue of overflow, but a shift in the binary point can turn it into a truncation issue. ■