Pascal II: Operand Formats (2 of 4)

Revised: 12/5/84
Security: Everyone

Pascal II: Operand Formats (2 of 4)

==========

Integers:

Integers in UCSD Pascal are whole numbers between -32768 to +32767, inclusive. They are stored in one word (2 bytes). Negative integers are represented in "two's complement," which means that they appear to have positive values greater than 32767; the negative integer is arrived at by subtracting 2 ^ 16 (65536) from this positve value. Similarly, large positive integers are stored as a complementary negative numbers (cf. Integer BASIC). The sign bit (MSB) is 0 if positive, 1 if negative.

```
  <-------byte----------->    <-------byte--------->
  15  14  .  .  .  .  .  8    7  .  .  .  .  .  .  0   <== 16 bits
  Sign                   Integer Value
```

Example: the number 3 is represented in binary as:

```
        MSB 0  0  0  0  0  0  0  0    0  0  0  0  0  0  1  1 LSB
```

However, -3 is represented as:

```
        MSB 1  1  1  1  1  1  1  1    1  1  1  1  1  1  0  1 LSB
```

which also reads as 65533 (or 65536-3)!

Integers may be passed by value or as Var parameters.

Reals:

Real numbers, in UCSD Pascal, are floating point numbers between +/-1.17550E-38 to +/-3.40282E+38, inclusive. Real numbers take up four bytes (2 words) of storage. Their binary representations are similar to the proposed IEEE standard for floating point numbers:

```
    31   30 . . . . . 23  22 . . . . . . . . . 0  <== 32 bits
   Sign      Exponent              Mantissa
```

"Mantissa" is the name given to the decimal portion of a number; by convention, it's expressed in scientific (exponential) notation. The "exponent" indicates the power to which the mantissa is raised. The exponent is represented in base 2 (2^n). The number 3 x 10^2, for instance, is defined as having a mantissa of 3, an exponent of 2, in base 10 (decimal).

The sign bit refers to the sign of the mantissa; it's 0 if positive, 1 if negative. The exponent is "offset" by 127; that is, a value of 127 in the exponent field corresponds to an exponent of 0. Similarly, if the value is 1, the exponent is -126, and if the field is 254, the exponent is +127. A value of 0 indicates that the real number is 0.

The mantissa of the real number is stored in normalized format in bits 0-22. "Normalizing" a number means adjusting it so that the highest bit is significant (that is, set to 1). The exponent indicates how many times (and in which direction) the value was shifted during normalization.

Notice that the MSB of the mantissa of any non-zero number that has been normalized is always a one. Zero can be treated as a special case: the exponent is simply set to zero. So, to gain additional precision, the mantissa has an implied "1" that is not stored, resulting in a functional 24-bit mantissa, even though only 23 bits are actually used. This gives slightly more than a 6-decimal-place (single precision) accuracy.

To make this clearer, let's look at some examples:

```
   Real number = 1
    MSB 0     01111111      00000000000000000000000 LSB
    Exponent = 127 (2^0) Mantissa = 1 (the implied 1 isn't stored)

   Real number = -9.9
    MSB 1     10000010      00111100110011001100110 LSB
    Exponent = 130 (2^3) Mantissa = 99000015
```

In the second example, the real number (in binary) appears as 1001.1110011... During normalization, the decimal point is moved to the left 3 times (incrementing the exponent), and the most significant bit becomes implied. The sign bit is 1, indicating that the number is negative.

Real numbers may be passed by value, or else they may be defined as Var parameters and then passed by address.

Apple Tech Notes

Tech Info Library Article Number:687