Bit-String Flicking

Bit strings (binary numbers) are frequently manipulated using logical operators, shifts, and circulates. Mastering this topic is essential for systems programming, programming in assembly language and optimizing code.

A typical use of bit string is to maintain a set of flags. Suppose that associated with a data structure in a program are 10 "options", each of which can be either "on" or "off". One could maintain this information using an array of size 10, or one could use a single variable (if it is internally stored using at least 10 bits, which is usually the case) and use 10 bits to record each option. In addition to saving space - a significant consideration if enough data structures and options are involved - the program is often cleaner if a single variable is involved rather than an array. Incidentally, bit strings are usually used to maintain Pascal "sets".

The logical operators which will be used are: AND(&), OR(|), XOR(⊕) and NOT(~). These operators examine the operand(s) on a bit by bit basis. For example, (10110 AND 01111) has a value of 00110. The AND, OR and XOR are binary operators; the NOT is a unary operator. The category description of Boolean Algebra/Digital Electronics has a complete description of each logical function. The following chart summarizes this information:

p	q	p AND q	p OR q	p XOR q	NOT p
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Logical shifts (LSHIFT-*x* and RSHIFT-*x*) "ripple" the bit string *x* positions in the indicated direction. Bits shifted out are lost; zeros are shifted in at the other end. Circulates (RCIRC-*x* and LCIRC-*x*) "ripple" the bit string *x* positions in the specified direction. As each bit is shifted out one end, it is shifted in at the other end. Thus, for this category, the size of a bit string is fixed; it cannot be lengthened or shortened by any of the logical operators, shifts or circulates. If any bit strings are initially of different lengths, all shorter ones are padded with zeros in the left bits until all strings are of the same length. The following table gives some examples of these operations:

p	LSHIFT-2 p	RSHIFT-2 p	LCIRC-3 p	RCIRC-3 p
01101	10100	00011	01011	10101
10	00	00	01	01
1110	1000	0011	0111	1101
1011011	1101100	0010110	1011101	0111011

The order of precedence (from highest to lowest) is: NOT; SHIFT and CIRC; AND; XOR; and finally, OR. Operators with equal precedence are evaluated left to right; all operators bind from left to right.

(NOT 10011) AND 01001 OR (01110 AND 01111)

01110

(LSHIFT-1 01101) AND (RCIRC-2 11100)

00010

((RCIRC-14 (LCIRC-23 01101)) | (LSHIFT-1 10011) & (RSHIFT-2 10111))

10110

(LSHIFT -2 (LCIRC- 2 (RSHIFT -1 10100)))

00100

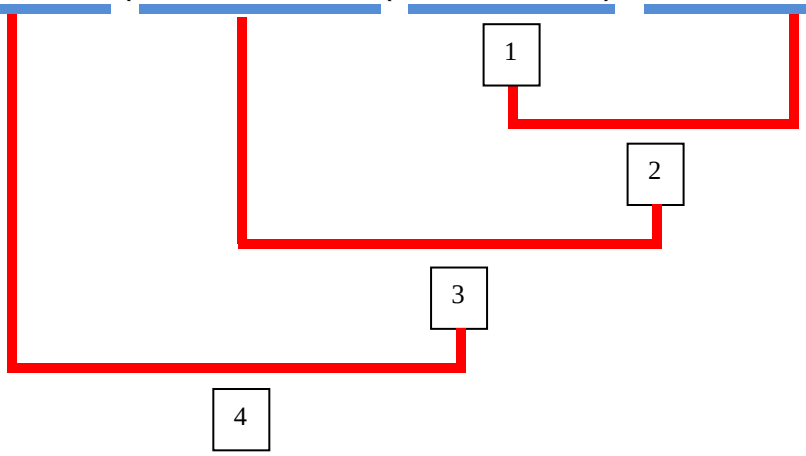
01101 OR (NOT 10111) AND 10010

01101

(LCIRC-2 01110) OR ((NOT 10110) AND (RSHIFT-1 01110))

11001

(LSHIFT-1 (10110 XOR (RCIRC-3 X) AND 11011)) = 01100



X = abcde

1 cdeab

2 cdeab AND 11011 = cd0ab

3 cd0ab XOR 10110 = Cd1Ab

4 d1Ab0 = 01100

a = 0

b = 0

c = ?

d = 0

e = ?

00?0?

00000

00100

00001

00101