## Personal Computer World Benchmark Tests

### Benchmark 1

20 FOR k=1 TO 1000  
40 NEXT k

FOR-NEXT loops are one of the commonest constructs in BASIC and if they are slow then all programs are likely to be slow.

### Benchmark 2

20 LET k=0   
30 LET k=k+1  
50 IF k<1000 THEN GOTO 30

This has the same effect as benchmark 1 of having the variable k count from 1 to 1000, but with explicit addition and a jump back. This method is generally much slower than the optimised FOR-NEXT loop. Notice that the benchmarks use line numbers and GOTOs. Most BASICs of the period lacked more sophisticated loop constructs.

### Benchmark 3

20 LET k=0  
30 LET k=k+1  
40 LET a=k/k\*k+k-k  
50 IF k<1000 THEN GOTO 30

This time some calculations are done inside the loop. This tests the speed of the arithmetic operators and also of accessing variables. For the benchmarks all variables are assumed to be of floating-point type.

### Benchmark 4

20 LET k=0  
30 LET k=k+1  
40 LET a=k/2\*3+4-5  
50 IF k<1000 THEN GOTO 30

The same as benchmark 3 except that constants are used for the arithmetic instead of a variable. If the code for looking up values of variables is inefficient then this test will be faster than the previous one.

### Benchmark 5

20 LET k=0  
30 LET k=k+1  
40 LET a=k/2\*3+4-5  
45 GOSUB 700  
50 IF k<1000 THEN GOTO 30   
700 RETURN

This introduces a subroutine call. It requires the return address to be put on a stack and then the destination of the call to be found, usually by searching through from the beginning of the program. Long programs in early versions of BASIC would make extensive use of subroutines and thus the efficiency of the calling mechanism was important.

### Benchmark 6

20 LET k=0  
25 DIM m(5)  
30 LET k=k+1  
40 LET a=k/2\*3+4-5  
45 GOSUB 700  
46 FOR l=1 TO 5  
48 NEXT l  
50 IF k<1000 THEN GOTO 30  
700 RETURN

This defines a small array at the start and adds another FOR-NEXT loop inside the main loop. This benchmark has little importance on its own but is used as a baseline for benchmark 7.

### Benchmark 7

20 LET k=0  
25 DIM m(5)  
30 LET k=k+1  
40 LET a=k/2\*3+4-5  
45 GOSUB 700  
46 FOR l=1 TO 5  
47 LET m(l)=a  
48 NEXT l  
50 IF k<1000 THEN GOTO 30  
700 RETURN

This assigns a value to each of the array elements every time through the loop. How much slower this is than benchmark 6 indicates the efficiency of array access.

### Benchmark 8

20 LET k=0  
25 DIM m(5)  
30 LET k=k+1  
40 LET a=k^2  
45 LET b=LN(k)  
47 LET c=SIN(k)  
50 IF k<1000 THEN GOTO 30

The final benchmark uses the more advanced mathematical functions of raising to a power (usually implemented using logarithms), calculating natural logarithms and calculating trigonometric sines.