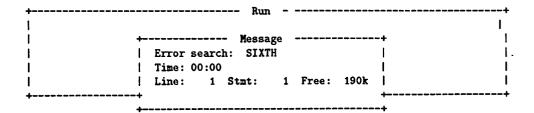
## FIVE FRIGHTENING FACTS ABOUT FLOATING-POINT ARITHMETIC

W. Kahan May 3, 1988

- 1. WHAT YOU SEE IS NOT NECESSARILY WHAT YOU GET.
- 2. WHAT YOU GET IS NOT NECESSARILY WHAT YOU EXPECT.
- 3. MANY A DISCREPANCY BETWEEN WHAT YOU GET AND WHAT YOU EXPECT IS MOST UNLIKELY EVER TO BE UNDERSTOOD, MUCH LESS CORRECTED, EVEN THOUGH IN PRINCIPLE EVERY DISCREPANCY CAN BE CORRECTED.
- 4. SIGNIFICANT DISCREPANCIES ARE VERY RARE, TOO RARE TO WORRY ABOUT ALL THE TIME, YET NOT RARE ENOUGH TO IGNORE.
- 5. THE DISCREPANCIES THAT ALREADY AFFLICT FLOATING-POINT ARITHMETIC DO NOT GRANT THE SYSTEM DESIGNER CARTE BLANCHE TO ADD A FEW MORE OF HIS OWN.

```
----- Borland Turbo Basic on an IBM PC -----
 ------ Edit
  CLS
  q = 3.0/7.0
  Print " The value of q = "; q
  Print "
          but 3.0/7.0 = "; 3.0/7.0
  Print
  Print " What You See Is Not Necessarily What You Get."
  ----- Run -----
  The value of q = .4285714328289032
   but 3.0/7.0 = .4285714285714286
  What You See Is Not Necessarily What You Get.
-----+
  CLS
  z = 0 : Print Using " z = \frac{\pi}{\pi}"; z
  y = 0.000123 : Print Using " y = \#\#\#.\#\#"; y
  x = y/100: Print Using " x = \frac{\#}{\#}. \frac{\#}{\#}; x
```

```
Print Using "y/x = \#\#\#.\#\#"; y/x |
     Print
     Print " What You See Is Not Necessarily What You Get."
  ----- Run ----
     z = 0.000
     y = 0.000
     x = 0.000
  y/x = 100.000
 What You See Is Not Necessarily What You Get.
        WHAT YOU SEE IS NOT NECESSARILY WHAT YOU GET.
----- Borland Turbo Basic on an IBM PC -----
CLS
  p = 2: for i=1 to 6: p = p*p: next i
  pp1 = p + 1 : pm1 = p - 1
   d = pp1 - pm1
   Print " We expect d = 2 , but actually d = "; d ; " ,"
   Print " although (p+1) - (p-1) = "; (p+1) - (p-1) ; " . !"|
   Print
   Print " What you get isn't necessarily what you expected." |
     ----- Run ------
    We expect d = 2, but actually d = 0,
    although (p+1) - (p-1) = 1 . !
    What you get isn't necessarily what you expected.
    WHAT YOU GET ISN'T NECESSARILY WHAT YOU EXPECTED.
                  ------- Edit ------
  | p0K = ((((2.0\^2)\^2)\^2)\^2)
  | pBAD = (((((2.0\^2)\^2)\^2)\^2)\^2 ' <<< Error 5: Illegal function call |
  | Print "This Error is really caused by misuse of the 8087's stack."
```



YOU ARE UNLIKELY EVER TO UNDERSTAND, MUCH LESS TO REMEDY EVERY ANOMALY.

----- Borland Turbo Basic on an IBM PC -----

```
-----+
| for i = 1 to 8000 : tj = 1
| for j = 0 to 15
                : \quad \texttt{tk} = \mathbf{1}
    for k = 0 to j
      d = tj + tk ' ... = 2\hat{j} + 2\hat{k} 1 0
      q = i/d : x = q*d ' <----<< 1 1
IF NOT(x=i) THEN ' <----<< 2 0
        print "x = ";x;" NOT = ";i '<----<< 2 1
        STOP: END IF ' <----< 2 2 8 |
      tk = tk+tk' \dots = 2^{(k+1)}
                                   309
                                   3 1 10 l
      next k
                                    3 2 12 |
    tj = tj+tj, ... = 2\^(j+1)
    next j : next i '
                                    3 3 16
 print " x = i ALWAYS !" '... and d = 17, 18, 20, 24, 32,
                     '... 33, 34, 36, 40, 48, ...
   -----+
  x = i ALWAYS!
```

## A PECULIAR PROPERTY OF DIVISION AND MULTIPLICATION WHEN ROUNDED ACCORDING TO IEEE STANDARD 754

This very simple program is certain to stop prematurely if the computer's floating-point arithmetic uses any other radix than 2 (binary), and almost certain to stop prematurely if division or multiplication is not rounded according to the IEEE standard. It stops prematurely on IBM 370s, DEC VAXs, CDC Cybers, CRAYs, all decimal calculators, ... But it says " x = i ALWAYS ! " on IBM PCs that use an 80x87 math. coprocessor, on all Apples that use SANE, on all SUN IIIs, on the ELXSI 6400, ...

CAN YOU EXPLAIN THIS ?