BASIC language interpreters and subroutine placement

JOHN H. LINGLE and TIMOTHY WALKER

*Livingston College, Rutgers University
New Brunswick, New Jersey 08903*

Plummeting prices and improved capabilities are making microcomputers with built-in BASIC interpreters increasingly common in behavioral science laboratories. Consequently, numerous articles are beginning to appear suggesting ways of maximally utilizing their capabilities. Because of these machines' frequent use for on-line data collection, several articles have discussed alternate ways of programming timing loops (cf. Owings & Fiedler, 1979; Perera, 1979). For users who need less than millisecond resolution or prefer to program only in BASIC, Perera (1979) recently compared the resolution of three BASIC language timing subroutines for the TRS-80 microcomputer. While such comparisons are useful, the resolution of one of the routines will vary widely depending on where it is placed in a larger program. The subroutine in question, Perera's Program 1, is replicated in Figure 1 for comparison, together with his Program 2.

Program 1 is distinguished from Program 2 (and from a third routine listed by Perera, 1979) by the GOTO statement that is executed on each pass through the timing loop (cf. Line 40 of Program 1). Whenever the BASIC interpreters of the popular TRS-80, Apple II, and PET microcomputers encounter a GOTO or GOSUB branching instruction, they search for the branching address beginning with the first program line. Consequently, the farther into the program the address line appears, the longer the interpreter needs to find and execute it. When a branching statement is executed repeatedly (as in Program 1), large discrepancies in run time will occur, depending on where in a program the address is located.

As a simple illustration of this phenomenon, we ran Programs 1 and 2 on a TRS-80 microcomputer at the beginning, in the middle, and at the end of a 1,000-line program that contained only remark statements. As suggested by Perera (1979), the subroutines were run in each case for 100 sec and the number of iterations per second was computed (i.e., K/100). For more stable

Program 1	Program 2
10 K=0	10 FOR K=0 TO 10000
20 K=K+1	20 A\$=INKEY\$
30 A\$=INKEY\$	30 IF A\$="1" THEN (GOTO) 100
40 IF A\$="1" THEN GOTO 20	40 NEXT K
50 PRINT K/...	100 PRINT K/...

Figure 1.

Table 1
Mean Number of Iterations Completed Per Second as a
Function of Subroutine Program Position

Program Position	Program 1	Program 2
At the beginning	73.4	97.2
After 500 lines	14.6	96.5
After 1,000 lines	8.2	96.1

Note—Mean values are based on five 100-sec trials.

estimates, each test was repeated five times and a mean value was computed. The results are displayed in Table 1. Program 2, which relies on a stack-controlled FOR . . . NEXT loop and executes only a single branching instruction upon its completion, showed minimal resolution variation as a result of its program position. Program 1, on the other hand, completed 88% fewer iterations per second when it was placed at the end, as compared with the beginning, of the program. Thus, it is important when timing with this routine to be careful of placement. Even more important is the general lesson on program structure. One often encounters subroutines at the end of programs. If a subroutine is called repeatedly, as is very often the case, its improper placement can greatly retard program execution speed. As a rule, when using a BASIC language interpreter, the more frequently a subroutine is called, the closer it should be placed to the beginning of a program.

REFERENCES

- OWINGS, R. A., & FIEDLER, C. A. Measuring reaction time with millisecond accuracy using the TRS-80 microcomputer. *Behavior Research Methods & Instrumentation*, 1979, 11, 589-591.
PERERA, T. B. A comparison of BASIC language timing loops for the TRS-80 microcomputer. *Behavior Research Methods & Instrumentation*, 1979, 11, 592.

(Accepted for publication July 10, 1981.)