CHAPTER 7: EFFICIENCY AND INSTRUMENTATION

Machines have become increasingly cheap compared to people; any discussion of computer efficiency that fails to take this into account is shortsighted. "Efficiency" involves the reduction of overall cost — not just machine time over the life of the program, but also time spent by the programmer and by the users of the program.

A clean design is more easily modified as requirements change or as more is learned about what parts of the code consume significant amounts of execution time. A "clever" design that fails to work or to run fast enough can often be salvaged only at great cost. Efficiency does not have to be sacrificed in the interest of writing readable code — rather, writing readable code is often the only way to ensure efficient programs that are also easy to maintain and modify.

To begin, let us state the obvious. If a program doesn't work, it doesn't matter how fast it runs. For instance:

```
C     PAYROLL COMPUTATION PROGRAM
C     READ EMPLOYEES ID,HOURS WORKED AND PAY RATES
   10 READ(5,20) EMPID,HOURS,SRATE,ORATE
   20 FORMAT(2F5.0,2F5.2)
      IF(EMPID .EQ. 77777.0)GO TO 60
      IF(HOURS .GT. 40.0)GO TO 50
C     COMPUTE WEEKLY PAY
      WAGE = SRATE * HOURS
C     CONVERT EMPLOYEES ID TO FIXED POINT FOR PRINTOUT
      IEMPID = EMPID
C     PRINT EMPLOYEES ID AND WAGE
   30 WRITE(6,40) IEMPID,WAGE
   40 FORMAT(12H1EMPLOYEE ID,I5,7H   WAGE,F8.2)
      GO TO 10
C     COMPUTE OVERTIME PAY
   50 A = ORATE * (HOURS - 40.0)
C     COMPUTE BASE PAY
      B = SRATE * 40.0
C     COMPUTE TOTAL WAGE
      WAGE = A + B
      GO TO 30
   60 STOP
      END
```

After we unscramble the confusing flow of control, we can see that the integer version of the employee's ID, IEMPID, is not set correctly when there is overtime — its value is a leftover from the last employee who had no overtime. The cause is