In More Depth: Using Hardware-Independent Metrics

Because accurately predicting and comparing performance is so difficult, many designers and researchers have tried to devise methods to assess performance that do not rely on measurements of execution time. These methods are frequently employed when designers compare different instruction sets to factor out the effects of different implementations or software systems and arrive at conclusions about the performance obtainable for different instruction sets.

One such method, which has been used in the past, is to use code size as a measure of speed. With this method, the instruction set architecture with the smallest program is fastest. The size of the compiled program is, of course, important when memory space is at a premium, but it is not the same as performance. In fact, today, the fastest computers tend to have instruction sets that lead to larger programs but can be executed faster with less hardware.

Evidence of the fallacy of using code size to measure speed can be found on the cover of a book published in 1973, shown in Figure 4.8.5. The figure clearly shows the lack of a direct relationship between code size and execution time. For example, the CDC 6600's programs are almost twice as big as those on the Burroughs B5500, yet the CDC computer runs Algol 60 programs almost six times *faster* than the B5500, a computer designed specifically for Algol 60. (The measurements for the B5500 and CDC 6600 are shown as dashed lines in Figure 4.8.5.)

Compiler writers sometimes use code size to choose between two different code segments on the same architecture. While this is less misleading than trying to compare code size across architectures, the accuracy of predicting performance from code size can vary widely.

4.34 [5] <§§4.2, 4.3> The VAX was Digital Equipment Corporation's 32-bit architecture. It was designed on the basis of many abstract principles: uniform addressing to all objects whether in memory or registers; extension of all operations to 8-, 16-, and 32-bit objects; and powerful high-level operations that could replace loops in other architectures. In the abstract, the VAX was the ideal architecture. In practice, despite lower instruction counts, implementations of the VAX had high values of CPI and often had lower clock rates than implementations of simpler architectures. Assume a simple architecture requires 1.5 times as many instructions for a program as a VAX, but a processor, P, implementing that architecture still runs 5 times faster than a VAX implementation, V. If the CPI of P is 1.5 for that program, what is the CPI of V for the program? Assume P and V have the same clock rate.

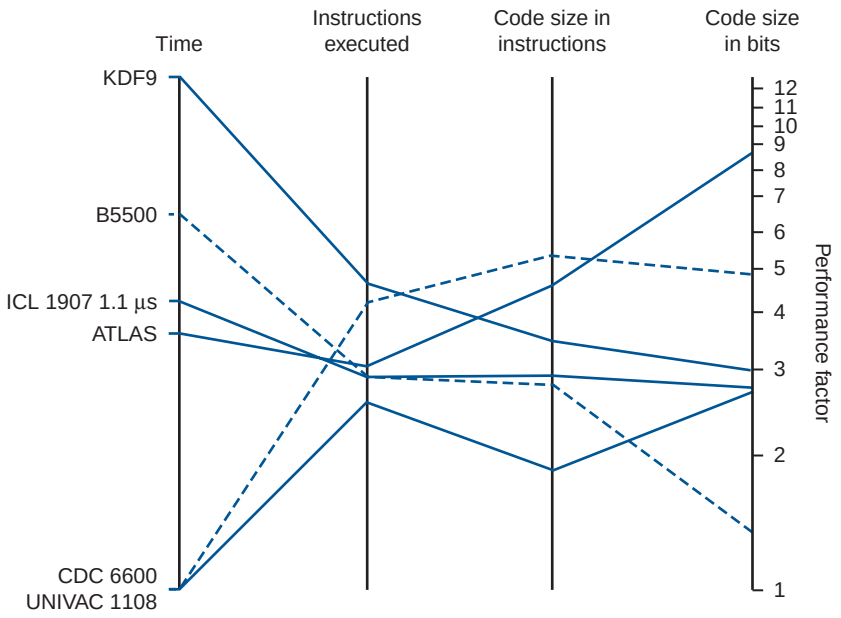

FIGURE 4.8.5 This graph is from the cover of Algol 60 Compilation and Assessment by B. A. Wichmann (published in 1973). The graph shows relative execution time, instructions executed, and code size in both instructions and bits for a set of programs written in Algol 60 and run on six different computers. The results are normalized to a reference computer, with a higher number indicating more execution time, larger instruction counts, or larger code size. The graph clearly shows that abstract measures such as code size bear little relationship to performance measures such as execution time or instructions executed. Despite the evidence that code size and execution time could be totally unrelated, many designers continued to emphasize code size throughout the 1980s. Graph redrawn with permission from Academic Press.