## 4.7 Historical Perspective and Further Reading

From the earliest days of computing, designers have specified performance goals—ENIAC was to be 1000 times faster than the Harvard Mark-I, and the IBM Stretch (7030) was to be 100 times faster than the fastest computer then in existence. What wasn't clear, though, was how this performance was to be measured.

The original measure of performance was the time required to perform an individual operation, such as addition. Since most instructions took the same execution time, the timing of one was the same as the others. As the execution times of instructions in a computer became more diverse, however, the time required for one operation was no longer useful for comparisons.

To take these differences into account, an *instruction mix* was calculated by measuring the relative frequency of instructions in a computer across many programs. Multiplying the time for each instruction by its weight in the mix gave the user the *average instruction execution time*. (If measured in clock cycles, average instruction execution time is the same as average CPI.) Since instruction sets were similar, this was a more precise comparison than add times. From average instruction execution time, then, it was only a small step to MIPS. MIPS had the virtue of being easy to understand; hence it grew in popularity. The In More Depth section on this CD discusses other definitions of MIPS, as well as its relatives MOPS and FLOPS!

### The Quest for an Average Program

As processors were becoming more sophisticated and relied on memory hierarchies (the topic of Chapter 7) and pipelining (the topic of Chapter 6), a single execution time for each instruction no longer existed; neither execution time nor MIPS, therefore, could be calculated from the instruction mix and the manual. While it might seem obvious today that the right thing to do would have been to develop a set of real applications that could be used as standard benchmarks, this was a difficult task until relatively recent times. Variations in operating systems and language standards made it hard to create large programs that could be moved from computer to computer simply by recompiling. Instead, the next step was benchmarking using synthetic programs. The Whetstone synthetic program was created by measuring scientific programs written in Algol 60 (see Curnow and Wichmann's [1976] description). This program was converted to Fortran and was widely used to characterize scientific program performance. Whetstone performance is typically quoted in Whetstones per second—the number of executions

of one iteration of the Whetstone benchmark! Dhrystone was developed much more recently (see Weicker's [1984] description and methodology).

About the same time Whetstone was developed, the concept of *kernel benchmarks* gained popularity. Kernels are small, time-intensive pieces from real programs that are extracted and then used as benchmarks. This approach was developed primarily for benchmarking high-end computers, especially supercomputers. Livermore Loops and Linpack are the best-known examples. The Livermore Loops consist of a series of 21 small loop fragments. Linpack consists of a portion of a linear algebra subroutine package. Kernels are best used to isolate the performance of individual features of a computer and to explain the reasons for differences in the performance of real programs. Because scientific applications often use small pieces of code that execute for a long period of time, characterizing performance with kernels is most popular in this application class. Although kernels help illuminate performance, they often overstate the performance on real applications. For example, today's supercomputers often achieve a high percentage of their peak performance on such kernels. However, when executing real applications, the performance often is only a small fraction of the peak performance.

## SPECulating about Performance

An important advance in performance evaluation was the formation of the System Performance Evaluation Cooperative (SPEC) group in 1988. SPEC comprises representatives of many computer companies—the founders being Apollo/Hewlett-Packard, DEC, MIPS, and Sun—who have agreed on a set of real programs and inputs that all will run. It is worth noting that SPEC couldn't have come into being before portable operating systems and the popularity of high-level languages. Now compilers, too, are accepted as a proper part of the performance of computer systems and must be measured in any evaluation.

History teaches us that while the SPEC effort may be useful with current computers, it will not meet the needs of the next generation without changing. In 1991, a throughput measure was added, based on running multiple versions of the benchmark. It is most useful for evaluating timeshared usage of a uniprocessor or a multiprocessor. Other system benchmarks that include OS-intensive and I/O-intensive activities have also been added. Another change, motivated by a problem discussed in the In More Depth section, was the decision to drop some benchmarks and add others. One result of the difficulty in finding benchmarks was that the initial version of the SPEC benchmarks (called SPEC89) contained six floating-point benchmarks but only four integer benchmarks. Calculating a single summary measurement using the geometric mean of execution times normalized to a VAX-11/780 meant that this measure favored computers with strong floating-point performance.

In 1992, a new benchmark set (called SPEC92) was introduced. It incorporated additional benchmarks, dropped matrix300, and provided separate means (SPECINT and SPECFP) for integer and floating-point programs. In addition, the SPECbase measure, which disallows program-specific optimization flags, was added to provide users with a performance measurement that would more closely match what they might experience on their own programs. The SPECFP numbers show the largest increase versus the base SPECFP measurement, typically ranging from 15% to 30% higher.

In 1995, the benchmark set was once again updated, adding some new integer and floating-point benchmarks, as well as removing some benchmarks that suffered from flaws or had running times that had become too small given the factor of 20 or more performance improvement since the first SPEC release. SPEC95 also changed the base computer for normalization to a Sun SPARCstation 10/40, since operating versions of the original base computer were becoming difficult to find!

The most recent version of SPEC is SPEC2000. What is perhaps most surprising is that SPEC2000, while having more programs than any earlier version of the SPEC benchmarks, has only slight overlap with SPEC95, and essentially none with the 1989 and 1992 releases. The next release of SPEC CPU should be in 2004 or 2005.

SPEC has also added additional benchmark suites beyond the original suites targeted at CPU performance. In 2003, SPEC provides benchmark sets for graphics, high-performance scientific computing, object-oriented computing, file systems, Web servers and clients, Java, and engineering CAD applications.

Creating and developing such benchmark sets has become difficult and time consuming. Although SPEC was initially created as a good-faith effort by a group of companies, it became important to competitive marketing and sales efforts. The selection of benchmarks and the rules for running them are made by representatives of the companies that compete by advertising test results. Conflicts between the companies' perspectives and those of consumers naturally arise. Perhaps in the future the decisions about such performance benchmarks should be made by, or at least include, a more representative group.

## Further Reading

Curnow, H. J., and B. A. Wichmann [1976]. "A synthetic benchmark," *The Computer J.* 19 (1):80.

*Describes the first major synthetic benchmark, Whetstone, and how it was created.*

Flemming, P. J., and J. J. Wallace [1986]. "How not to lie with statistics: The correct way to summarize benchmark results," *Comm. ACM* 29:3 (March) 218–21.

*Describes some of the underlying principles in using different means to summarize performance results.*

McMahon, F. M. [1986]. "The Livermore FORTRAN kernels: A computer test of numerical performance range," Tech. Rep. UCRL-55745, Lawrence Livermore National Laboratory, Univ. of California, Livermore (December).

*Describes the Livermore Loops—a set of Fortran kernel benchmarks.*

Smith, J. E. [1988]. "Characterizing computer performance with a single number," *Comm. ACM* 31:10 (October) 1202–06.

*Describes the difficulties of summarizing performance with just one number and argues for total execution time as the only consistent measure.*

SPEC [2000]. *SPEC Benchmark Suite Release 1.0,* SPEC, Santa Clara, CA, October 2.

*Describes the SPEC benchmark suite. For up-to-date information, see the SPEC Web page via a link at www.spec.org.*

Weicker, R. P. [1984]. "Dhrystone: A synthetic systems programming benchmark," *Comm. ACM* 27:10 (October) 1013–30.

*Describes the Dhrystone benchmark and its construction.*