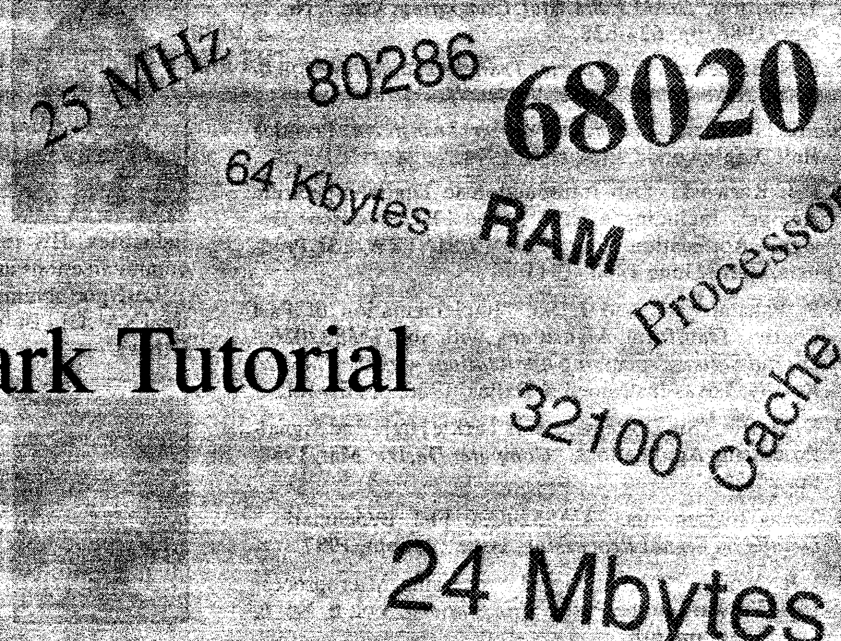

A Benchmark Tutorial

What do benchmark test results really mean? Here's how to evaluate the data in terms of your own application.

Walter J. Price
Motorola

Simply put, a benchmark is a standard for judging relative performance among various computers. Unfortunately, any simplicity ends with this definition. The first problem arises from the fact that users have no official standard to follow when they want to evaluate a benchmark. The second problem revolves around the truism that the best benchmark in the world that measures someone else's application does just that. From the user's perspective, the best benchmark in the world accurately measures system performance in a target application. The task of creating a good benchmark includes the determination of which tests are applicable to the user's environment and how to determine the results.

Here I generally describe benchmarking and discuss some of the specific benchmark tests for computer systems that are in use today. I also examine some of the pitfalls involved with benchmark comparison and analysis and point out how to avoid them—or at least to minimize the impact of such problems. The goal is to learn how to gather and interpret meaningful comparison data.

Benchmarking caveats

Producing and interpreting benchmark data crosses into the realm between art and science. No single-task benchmark test (one that only measures one aspect of a computer's performance) can fully characterize the true performance capability of a system under actual user loads. Safety does reside in numbers. A collection of different benchmark tests merely provides some averages. But users must take care when comparing the results.

Features like optimizing options on today's compilers can drastically affect the results of benchmark tests. Some vendors publish a range of test results using different optimization levels. This documentation probably provides the most reliable form of public-domain benchmark data. Combining this data with the appropriate system configurations that were actually used in the tests provides the background information needed to reproduce the data, if necessary.

0272-1732/89/1000-0028$1.00 © 1989 IEEE

Users should also consider the source or "pedigree" of a benchmark. Benchmark code tends to migrate quite freely, given public-domain networks like Usenet. Modifications made to this code, for whatever reason, cause the evolution of many different versions. Hand-coded libraries or conversions from one language to another can cause successive generations of the test. Results vary according to which version of a benchmark test is used. Because of this process, some benchmarks have degenerated to the point of being virtually useless.

Users should avoid making a determination of overall system performance based on one benchmark test. Many benchmarks are one-dimensional in nature (that is, they test only one aspect of a system). Some popular tests stress only raw processor-instruction bandwidth or floating-point performance. Other items affecting system performance include

- file system and compiler efficiencies,
- cache-memory size and organization,
- main-memory size and organization,
- I/O and file transfer speeds,
- user loads, and
- application mixes.

Most benchmark tests do not verify the validity of the performance results they produce. Without verified test results, users can find it difficult to determine whether the benchmark was properly performed. Some optimizing compilers recognize small benchmark suites and simply return the expected result without performing any computation. Totally meaningless performance data results.

The only way to truly compare benchmark test results is to port and compile the same copy of the operating-system source to all of the systems under consideration. This procedure minimizes the effects from differences in binary copies of the code and ensures a common base of software for these systems. Since this task is beyond the reach of most organizations, users can employ another option. Setting up the test carefully and obtaining a clear understanding of how to interpret the results can yield a successful comparison. The goal is to establish a level playing field for all of the systems being tested.

Some additional questions to ask when analyzing or generating benchmark data follow.

1) Which real-world application does the benchmark measure, and how does the user accurately characterize the system's performance under typical application work loads?

Try to choose a set of benchmark tests that analyzes the components critical to producing optimal user performance. For example, a CPU-intensive benchmark by itself does not provide sufficient data to interpret how a system will perform in an I/O-intensive environment like transaction processing. In this example, the benchmark suite should also measure disk-I/O and file-system efficiencies and work-load capacity (keystroke-handling capability).

2) What factors influenced the generation of the benchmark data?

Correlating data from different or biased sources probably will not produce an accurate comparison. A vendor may provide data from a "hot box" (or nonproduction hardware) or use a compiler that "recognizes" a benchmark suite and loads a hand-optimized algorithm for the test. Valid data should contain documentation of test configurations and optimization levels. The tests should be repeatable using a production-grade system.

The benchmark test-result data that appears in this article (shown in tables) constitutes a general guideline only. Users should implement application-specific tests (that is, actual application software running in a typical user environment) to supplement this information.

3) Were devices like cache accelerators or optimizing compilers used in all of the configurations tested?

Cache accelerators can make a small benchmark run much faster than if the same code were to run in another memory location. Optimizing compilers can virtually reduce a repetitive benchmark to a simple NOP (no operation) and yield totally meaningless results.

4) Was one system tuned for a particular benchmark?

Some vendors have a reputation for publishing benchmark data obtained from a hot box or a modified system that a customer could not buy or duplicate with off-the-shelf components. Simple operating-system kernel tuning can improve benchmark results by as much as 30 percent. Hand-optimized utilities can improve benchmark performance by as much as 30 percent. All operating-system parameters and compiler options should match closely from one system to another.

5) What sources should users employ for benchmark data?

To assure the best comparison of results, users should investigate several different sources. When possible, they should use the results from the same test suite on all systems being evaluated. Users can also employ independent third-party benchmark data as a verified, unbiased source of information.

In summary, users should

- determine which aspects of system or component performance are to be measured,
- determine the best source of benchmark suites or

performance data (either public-domain or licensed third-party packages),

- ensure that all system-hardware and operating-system parameters during benchmark comparisons equate as closely as possible, and
- understand what specific benchmark tests measure and what causes the results to vary.

Benchmark tests

Here I present a set of descriptions for many of the more popular benchmarks. I obtained the actual benchmark data contained in the following tables from a variety of public-domain sources. I do not intend these benchmark figures to provide definitive results, but show them to give a general indication of relative performance among various computers. In many instances, a published range of performance values exists for various computers. Since space cannot reasonably present the total data, I list only the highest and lowest benchmark results for a system in terms of its particular measurement.

I show the system model number and the general test configuration (processor type, the typical amount of cache and random-access memory available, the processor speed in megahertz, and the processor rating in millions of instructions per second) for each entry. (See the accompanying box for a discussion of MIPS.) This data provides a basic understanding of the test setup. As stated, obtaining a description of the entire hardware and software test setup—along with the actual results—adds meaning to model-by-model comparisons.

Finally, some table entries contain additional information in the comments column that provides background data on how test results were obtained. Examples include the operating system (such as VMS for the VAX and AIX for the IBM computer), the compiler options (optimized, unoptimized), and any special hardware (68882 floating-point units, or FPUs, and floating-point accelerators, or FPAs) used in the test. Dashes indicate the information was not published with the test results, or is generally unavailable.

The industry uses a number of public-domain or licensed third-party benchmarks, some of which I discuss in the following sections. (For an outline of some popular proprietary benchmarks, see the accompanying box.)

Dhrystone. This synthetic (nonreal-world) benchmark measures processor and compiler efficiency by executing a "typical" set of integer calculations. These calculations include integer arithmetic, character/string/array manipulation, and pointers. Reinhold P. Weicker constructed the benchmark by using measured statistical data from actual user programs. The procedure does not use operating-system calls, I/O functions, or floating-point operations. The results provide a general measure of user-level integer performance. This benchmark contains little code that can be optimized by vector processor systems.[2]

Weicker wrote the original Dhrystone benchmark in the Ada programming language. Rick Richardson later rewrote it in the C language and posted it on the Usenet network. Results appear in Dhrystones per second. Users should exercise caution with the Dhrystone test (as with any benchmark test) because it does not always reflect how large user applications perform. Also, optimizing compilers can remove useless code from Version 1.1 of the benchmark, which improves the per-

MIPS: A meaningful measurement?

Millions of instructions per second, or MIPS, is a popular—though superficial—way to describe computer system performance. MIPS typically come from either measured data or calculated maximum performance. We usually compare system performance ratings in MIPS with that of a VAX 11/780, which is considered to be a 1-MIPS machine. On a family of systems with a common processor (like the Motorola 88000), MIPS ratings can help judge relative system integer performance. The ultimate apples-to-oranges phenomenon occurs when we compare MIPS ratings between two different architectures like RISCs and CISCs (reduced and complex instruction-set computers). The key points to remember about MIPS are

- instructions do not remain constant from processor to processor, and
- MIPS are only meaningful in the context of a single processor family.

To make sense out of MIPS ratings, users must first define the term *instruction*. A direct correlation does not always exist between the number of instructions being executed and the amount of actual work being accomplished. On a jovial note, the industry has come up with many colorful ways to use the MIPS acronym:

- meaningless indicators of performance for systems,
- meaningless information of performance for salesmen, or
- meaningless information from pushy salesmen.

Proprietary benchmarks

The following performance benchmarks—in contrast to the others in this article—have not entered the public domain. Users who wish to obtain more information should contact the companies directly.

Aim benchmarks. Aim Technology in Palo Alto, California, sells and maintains two suites of multiuser benchmark tests.

Suite III. Written in the C programming language, this suite simulates applications that fall into either task- or device-specific categories.

The task-specific routines simulate such functions as word processing, database management, and accounting. The device-specific code measures the performance of hardware features like memory, disk, floating-point, and I/O operations. All measurements represent a percentage of VAX 11/780 performance.

The performance and user ratings constitute the two most frequently quoted results. The performance rating represents a percentage of VAX 11/780 performance in which the VAX equals 100 percent (for example, the Motorola SYS3640 supermicrocomputer has a performance rating that is 400 percent of the VAX, or four times the performance).

The second parameter, or user rating, is the maximum number of concurrent users that a system can support. For example, a VAX 11/780 equals 12 users. In general, the Aim III suite gives a better overall indication of a system's performance than small, single-task benchmarks. The company verifies and maintains all official results. Published, copyrighted reports are available on individual computer systems.

Suite V. This benchmark suite measures throughput in a multitasking workstation environment. The design goals of this new suite include

- the ability to stress single-user, multitasking system performance,
- simulations of real-world systems that employ routines based on actual user applications,
- incremental system loading to gradually increase the stress on system resources, and
- testing multiple aspects of system performance.[1]

The graphically displayed results plot the workload level versus the amount of time (in seconds) to process the specific load level. Several different models characterize various user environments such as financial, publishing, and software development.

Business Benchmark. Developed and maintained by Neal Nelson and Associates in Chicago, this collection of 18 separate routines examines various aspects of system performance. The bulk of these routines consists of a series of loops that exercise functions such as disk I/O speed, floating-point performance, and processor and cache efficiency. Various combinations of the 18 routines characterize different real-world business applications like word processing, accounting, and application development. Unlike many other benchmark tests in use today, Business Benchmark results indicate that CISCs tend to outperform RISCs on some multiuser tasks.

formance results by as much as a factor of two. Dhrystone Versions 2.X eliminated this dead code to thwart the efforts of current optimizing compilers. Table 1 on the next page reflects Version 1.1.

Originally, Dhrystone solely used features like peephole optimizers, but today anything appears to be acceptable—short of in-line coding. When benchmark numbers are quoted, users should ask which compiler options were selected at execution time.

The Dhrystone routine itself contains some peculiar attributes that have been coded into the program. Part of the routine involves copying long, 30-character strings that happen to be on unaligned word boundaries. Dhrystone uses zero-offset address for 50 percent of its memory data references, where a more real-world number is somewhere between 10 and 15 percent.[2] These attributes tend to make some machines appear faster than others. The Dhrystone test seems to be lenient on processors like the Am29000 because the routine does not exercise CPU areas that would slow the 29000 down under typical user-application loads.[3] The Dhrystone benchmark is also small enough to fit into the instruction cache of some systems, which further skews the results.

Table 1.
Summary of Dhrystone 1.1 benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	Dhrystones/s Low	Dhrystones/s High	Comments
Alliant FX/8; MP=8	Proprietary	—	—	—	35.0	7,655	7,655	
ALR FlexCache 25386	80386	64	5	25.00	—	8,671	8,671	SCO Unix
Altos Series 2000	80386	—	—	16.00	3.0	4,237	4,348	
Amdahl 5860	—	—	—	—	—	28,846	28,846	C compiler, V. 1.22
Amdahl 5890/300E	—	—	—	—	—	43,668	43,668	C compiler
Apollo 5X0T	68020	—	—	20.00	3.4	6,250	6,250	
Apollo Series 10000	Prop. RISC	—	—	18.20	16.0	25,461	27,000	Up to four CPUs
Apollo Series DN3000	68020	—	—	12.50	1.2	2,186	2,186	
Apollo Series DN4000	68020	0	4	25.00	4.0	6,038	7,109	
Apple Macintosh	68000	—	—	7.70	—	625	625	
Apple Macintosh II	68020	—	—	15.70	—	2,106	2,719	H = Greenhills C compiler V. 1.8
Apple Macintosh Plus	68000	—	—	7.83	—	660	769	
AT&T 3B1	68010	—	—	10.00	—	973	1,033	Unix V. 1
AT&T 3B2/300	32000	—	—	7.20	—	409	699	L = Unix V. 2; H = Unix V. 3
AT&T 3B2/400	32100	6	4	10.00	1.0	672	1,120	L = Unix V. 3; H = Unix V. 2
CCI Power 5/32	68010	—	—	12.50	—	1,129	1,192	Unix 4.2 BSD
CCI Power 6/32	Proprietary	—	—	—	—	8,498	8,498	
CCI Power 7/64	Proprietary	—	—	—	—	53,108	53,108	
Compaq 386	80386	0	4	16.00	—	1,724	2,941	
Compaq 386/20	80386	0	4	20.00	—	7,575	9,335	
Compaq 386/25	80386	32	5	25.00	—	8,277	10,617	
Convergent Server PC	80386	64	4	20.00	5.7	6,534	9,436	L = unopt.; H = opt.
Convex C-1 XP 6.0	Proprietary	—	—	—	4.5	7,249	7,249	
Cray 1S	Proprietary	—	—	—	—	14,820	14,820	
Cray X-MP	Proprietary	—	—	—	—	18,530	18,530	
Data Gen. MV15000-20	Proprietary	16	64	11.80	8.0	8,300	8,300	
Data Gen. MV20000	Proprietary	16	64	—	10.0	8,300	8,300	
DEC μVAX 3500	Proprietary	—	—	11.10	3.5	4,746	4,746	
DEC μVAX II	Proprietary	—	—	5.00	0.9	1,326	1,612	L = VMS
DEC VAX 11/750	Proprietary	—	—	—	1.0	835	961	
DEC VAX 11/780	Proprietary	—	—	5.00	1.0	1,243	1,870	
DEC VAX 11/784	Proprietary	—	—	—	—	5,263	5,555	
DEC VAX 11/785	Proprietary	—	—	7.50	1.5	1,783	2,069	
DEC VAX 8550	Proprietary	—	—	—	6.4	8,000	10,416	
DEC VAX 8600	Proprietary	—	—	—	4.5	6,423	10,416	L = Unix 4.3 BSD
DEC VAX 8600	Proprietary	—	—	—	4.5	4,896	5,235	Ultrix V. 1.2
DEC VAX 8650	Proprietary	—	—	—	6.2	7,123	10,787	H = VMS
DEC VAX 8700	Proprietary	—	—	—	6.0	10,416	10,416	
DEC VAX 8810	Proprietary	—	—	22.22	12.0	10,416	10,416	
DEC Vaxstation 2000	Proprietary	—	—	5.00	0.9	1,502	1,502	
DEC Vaxstation 3200	Proprietary	—	—	5.00	2.0	5,271	5,271	
Decstation 3100	R2000	128	8	16.70	14.0	16,870	26,600	L = unopt.; H = opt.
Force CPU-21B	68020	—	—	25.00	—	5,555	5,555	
Force CPU-386A	80386	—	—	16.00	—	6,200	6,200	
HP 9000 Mod. 320	68020	—	—	16.70	2.0	2,464	2,671	HP/UX V. 5.02
HP 9000 Mod. 340	68030	—	—	16.70	—	6,536	6,536	
HP 9000 Mod. 360	68030	0	4-12	25.00	4.5	6,702	6,702	
HP 9000 Mod. 500	Proprietary	—	—	—	—	1,599	1,599	HP/UX V. 5.05; 1 processor
HP 9000 Mod. 500	Proprietary	—	—	—	—	3,020	3,020	HP/UX V. 5.05; 2 processors
HP 9000 Mod. 500	Proprietary	—	—	—	—	4,140	4,140	HP/UX V. 5.05; 3 processors
HP 9000 Mod. 550	Proprietary	—	—	—	—	1,518	1,531	HP/UX V. 5.11
HP 9000 Mod. 825S	Prop. RISC	16	—	12.50	3.0	17,829	17,829	
HP 9000 Mod. 825SRX	Prop. RISC	16	—	—	8.0	13,157	16,672	
HP 9000 Mod. 835S	Prop. RISC	128	—	15.00	4.0	23,430	23,441	
HP 9000 Mod. 835SRX	Prop. RISC	128	—	15.00	4.0	23,430	23,430	
HP 9000 Mod. 840	Prop. RISC	128	24	8.00	4.5	11,165	11,215	H = opt.

Table 1 (continued).

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	Dhrystones/s Low	Dhrystones/s High	Comments
HP 9000 Mod. 840S	Prop. RISC	128	24	8.00	4.5	9,920	9,920	
HP 9000 Mod. 850S	Prop. RISC	—	—	13.70	7.0	15,576	21,358	
IBM 3081	—	—	—	—	—	15,007	15,007	C compiler V. 1.5
IBM 3090/200	—	—	—	—	10.0	31,250	31,250	
IBM 4341 Mod. 12	Proprietary	—	—	14.70	—	3,690	3,910	Opt.
IBM 4381 Mod. 2	Proprietary	—	—	—	—	4,504	5,681	
IBM PC AT	80286	—	—	—	—	1,380	1,380	
IBM PC AT	80286	—	—	6.00	—	531	531	
IBM PC AT	80286	—	—	9.05	—	692	1,484	
IBM PS/2 Mod. 70-A21	80386	64	4	25.00	—	8,650	12,769	L = SCO Unix ; H = AIX
IBM RT PC	Prop. RISC	—	—	5.90	4.5	6,097	6,500	H = AIX
IBM RT PC Mod. 135	Prop. RISC	—	—	7.30	6.0	10,770	10,770	
Integr. Sol. Advantage2000	R2000	64	16	16.70	12.0	18,920	27,100	L = unopt.; H = opt.
Intel 386 ATS	80386	—	—	16.00	—	3,424	3,424	
Intergraph Interpro 32C	Clipper RISC	—	—	30.00	5.0	4,855	8,309	
Ironics IV-9001	Am29000	16	8	25.00	17.0	35,760	35,760	
MIPS M/1000	R2000	128	16	15.00	15.0	15,100	25,000	L = unopt.; H = opt.
MIPS M/120-3	R2000	128	8	12.50	10.0	23,300	23,300	
MIPS M/120-5	R2000	128	8	16.70	13.0	18,700	31,000	L = unopt.; H = opt.
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	30,700	47,400	L = unopt.; H = opt.
MIPS M/500	R2000	24	8	8.00	8.0	8,800	14,200	L = unopt.; H = opt.
MIPS M/800	R2000	128	8	12.50	12.5	12,800	21,300	L = unopt.; H = opt.
MIPS RC2030	R2000	64	16	16.70	12.0	19,100	31,200	L = unopt.; H = opt.
Motorola SYS1131	68020	16	2	16.70	1.5	3,246	3,257	
Motorola SYS1147	68030	0	4	20.00	3.8	6,334	6,334	
Motorola SYS2300	68020	0	4	16.70	1.5	4,876	4,876	
Motorola SYS2600	68020	16	4	16.70	1.5	4,566	4,566	
Motorola SYS3300	68030	0	4	20.00	3.8	6,334	6,334	
Motorola SYS3600	68030	0	4	25.00	4.7	8,826	8,826	
Motorola SYS3640	68030	64	4	25.00	5.3	7,942	8,900	
Motorola SYS3800	68030	64	—	33.00	6.9	9,239	11,000	H = Greenhills C V. 1.8.2
Motorola SYS8600	88100	32	8	20.00	17.0	35,714	35,714	
Multiflow Trace 7/200	—	—	—	—	—	14,195	14,195	
NCR Tower 32/400	68020	8	4	16.70	1.5	3,628	3,638	
NCR Tower 32/450	68020	8	4	25.00	—	4,941	4,941	
Opus Systems	88000	32	4-20	20.00	17.0	41,166	41,166	
Prime EXL 316	80386	—	—	16.00	3.2	7,112	7,112	Optimized
Pyramid 90x	Proprietary	—	—	8.00	2.5	1,779	3,333	High = w/cache
Pyramid 98x	Proprietary	—	—	10.00	5.4	3,627	3,856	
Silicon Graphics Iris	R2010	24	8	12.50	—	18,416	18,416	
Solbourne 4/600	Sparc	64	16	16.70	7.0	18,715	18,715	
Sparcstation 1	Sparc	0	4-24	20.00	12.5	22,049	22,049	
Sparcstation 330	Sparc	0	—	25.00	16.0	27,777	27,777	
Sun 3/160C w/68881	68020	0	4	16.70	2.0	2,800	3,850	L = unopt.; H = opt.
Sun 3/260 w/68881	68020	0	8-24	25.00	4.0	5,366	7,142	
Sun 3/470	68030	0	—	33.00	7.0	11,748	11,748	
Sun 3/50	68020	0	4	15.00	1.5	2,280	2,695	
Sun 3/60	68020	0	4	16.70	2.0	4,295	4,545	
Sun 3/80	68030	0	4	20.00	3.0	5,154	5,154	
Sun 386i Mod. 150	80386	32	8	14.30	3.5	8,388	8,388	w/80387 FPU
Sun 4/260 w/Weitek	Sparc	0	16-32	16.70	10.0	10,550	19,900	L = unopt.; H = opt.
Sun Sparc 4/110	Sparc	—	—	14.30	7.0	14,109	14,201	
Tandy 3000	80286	—	—	8.00	—	1,455	1,543	
Tandy 6000	68000	—	—	8.00	—	1,286	1,366	
Tektronix 4319	68020	0	4	20.00	2.5	7,581	7,581	
Unisys 5000/90	68020	—	—	12.50	1.0	3,307	3,311	

Table 2.
Summary of *Digital Review* benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	Seconds Low	Seconds High	Comments
Alliant FX/8	Proprietary	—	—	—	35.0	1.48	1.48	
Convex C-1 XP	Proprietary	—	—	—	4.5	0.487	0.487	
DEC μVAX II/GPX	Proprietary	—	—	5.00	0.9	9.17	9.17	
DEC VAX 11/780	Proprietary	—	—	5.00	1.0	6.75	6.75	
DEC VAX 8600	Proprietary	—	—	—	4.5	2.32	2.32	VMS V. 4.5
DEC VAX 8650	Proprietary	—	—	—	6.2	1.584	1.584	
DEC VAX 8700	Proprietary	—	—	—	6.0	1.469	1.469	
DEC Vaxstation 3200	Proprietary	—	—	5.00	2.0	2.9	2.9	
Elxsi 6420	Proprietary	—	16	20.00	—	1.193	1.193	
MIPS M/1000	R2000	128	16	15.00	15.0	0.94	0.99	
MIPS M/120-5	R2000	128	8	16.70	13.0	0.783	0.783	
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	0.553	0.553	
MIPS M/500	R2000	24	8	8.00	8.0	1.86	1.86	
MIPS M/800	R2000	128	8	12.50	12.5	1.2	1.2	
Sun 4/260	Sparc	0	16-32	16.70	10.0	1.72	2.09	Weitek FPU
Sun Sparc 4/110	Sparc	—	—	14.30	7.0	2.32	2.32	

Digital Review. *Digital Review* magazine has compiled a set of benchmark routines that mixes 34 individual integer and floating-point routines. The test itself stresses floating-point performance. This large benchmark contains over 3,000 lines of Fortran code. The *Digital Review* benchmark does not perform any verification of test results; these results usually appear as a list of the geometric mean of all tests, performed in seconds. On a secondary level, the test normalizes relative comparisons among various systems to the Digital MicroVAX II, which is equal to 1.0. These units are called MicroVAX units of processing (MVUPs). Table 2 lists only the raw benchmark-test results in seconds.

Users have criticized this benchmark for its odd structure and unusual instruction mix that does not accurately mimic real-world program flow. Initializing the routines within the timing loops, rather than running the actual benchmark code, consumes a large amount of time. This practice usually results in a low estimate of a system's actual MVUP rating. More recently, *Digital Review* magazine has taken steps to revise its benchmark (now called CPU2) to correct some of these odd programming sequences. (Table 2 reflects the previous version.)

Dodoc. This 5,300-line Fortran program—which simulates the operations within a nuclear reactor—accurately tests instruction-fetch bandwidth and scalar floating-point performance. Compilers can vectorize very little of the code. The routine uses the Monte Carlo method of simulation in which an iterative process converges on an expected result. The routine was originally designed as a check of both compiler and intrinsic (real-world) functions. Normalized results appear in terms of the ratio of CPU time needed to perform the test versus an arbitrary defined reference.[4] This *R* factor is normalized where 100 equals the performance of the IBM 370 Model 168. The algorithm calculates that $R =$ 48,671/seconds of processor time. Larger *R* factors equate to higher system performance.

Differences in floating-point accuracy (that is, single- or double-precision calculations), unique characteristics of mathematical libraries, and rounding errors contribute to how fast the algorithm converges on the expected answer. (See Table 3.)

Khornerstone. Developed by Workstation Laboratories, this benchmark yields a normalized rating on overall system performance using 22 separate tests.[5] This suite of tests includes a mix of both public-domain (Dhrystone, Sieve, etc.) and proprietary benchmark routines. The result is a unit of measure called Khornerstones per second. This set of routines measures characteristics of processor, floating-point, and disk performance. The Khornerstone test measures single-user

Table 3.
Summary of Dodoc benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	*R* Factor Low	*R* Factor High	Comments
Alliant FX/80; MP = 4	68020	—	32	—	—	248	248	64 bits
HP 9000 Mod. 370	68030	64	8-48	33.00	7.0	47	68	64 bits; L = 68882; H = Weitek
Alliant FX-1	Proprietary	—	—	12.00	—	85	85	64 bits
Alliant FX/8; MP = 8	Proprietary	—	—	—	35.0	101	101	
Amdahl 470 V8	—	—	—	—	—	150	150	64 bits
Amdahl 5860	—	—	—	—	—	475	475	64 bits
Apollo Series 10000	Prop. RISC	—	—	18.20	16.0	291	460	64 bits; H = Fortran V. 10.5r15
Bull DSP 90/x	—	—	—	—	—	371	371	64 bits
CCI Power 6/32	Proprietary	—	—	—	—	50	50	
CDC Cyber 990-E	Proprietary	—	—	—	6.2	592	592	64 bits
Celerity 1260	Proprietary	—	—	—	6.2	48	48	
Compaq 386/25	80386	32	5	25.00	—	73	73	64 bits; WTL3167 & DOS
Convex C-120	Proprietary	—	32	—	—	103	103	64 bits
Convex C-210; MP = 1	Proprietary	—	—	—	—	296	296	64 bits
Cray X-MP	Proprietary	—	—	—	—	1,080	1,080	
Cray X-MP/28	Proprietary	—	—	—	—	1,701	1,701	64 bits
Data Gen. MV20000	Proprietary	16	64	—	10.0	99	99	
Data Gen. MV40000	Proprietary	—	—	—	—	246	246	64 bits
DEC VAX 11/780 VMS	Proprietary	—	—	5.00	1.0	26	26	64 bits
DEC VAX 8600	Proprietary	—	—	—	4.5	91	91	VMS; 64 bits
DEC VAX 8650	Proprietary	—	—	—	6.2	129	129	
DEC VAX 8700	Proprietary	—	—	—	6.0	136	136	VMS; 64 bits
Decstation 3100	R2000	128	8	16.70	14.0	255	268	64 bits
Edge 1	Proprietary	—	—	—	—	53	53	
Floating-Pt. 264 SJE	Proprietary	—	—	—	—	396	396	64 bits
Fujitsu VP-200	—	—	—	—	—	915	915	64 bits
HP 9000 Mod. 835S	Prop. RISC	128	—	15.00	4.0	214	214	
HP 9000 Mod. 850S	Prop. RISC	—	—	13.70	7.0	201	201	
HP 9000 Mod. 855S	Prop. RISC	256	—	25.00	—	266	266	64 bits
Harris HCX-7	Proprietary	—	—	—	7.7	64	64	
IBM 3081G	—	—	—	—	—	181	181	
IBM 3081K	—	—	—	—	—	236	236	64 bits
IBM 3090	—	—	—	—	10.0	714	714	Scalar mode
IBM 3090/200	—	—	—	—	10.0	847	847	64 bits
IBM 4381 Mod. 2	Proprietary	—	—	—	—	90	90	64 bits
Intergraph Interpro C245	Clipper RISC	—	—	33.00	—	40	40	64 bits
Intergraph Interpro C370	Clipper RISC	—	—	40.00	—	72	72	64 bits
MIPS M/1000	R2000	128	16	15.00	15.0	227	227	
MIPS M/120-5	R2000	128	8	16.70	13.0	280	289	64 bits
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	438	443	64 bits
MIPS M/500	R2000	24	8	8.00	8.0	88	100	
MIPS M/800	R2000	128	8	12.50	12.5	190	19	
MIPS RC2030	R2000	64	16	16.70	12.0	238	238	64 bits
Silicon Graphics 4D/70	R2010	24	8	12.50	—	170	170	64 bits
Sun 3/110	68020	0	4	16.70	2.0	17	17	
Sun 3/160 W/FPA	68020	0	4	16.70	2.0	35	35	64 bits
Sun 3/260	68020	0	8-24	25.00	4.0	22	43	L = 68881; H = Weitek
Sun 3/260 w/FPA	68020	0	8-24	25.00	4.0	54	54	64 bits
Sun 386i Mod. 250	80386	32	16	25.00	—	26	26	64 bits
Sun 4/260 w/Weitek	Sparc	0	16-32	16.70	10.0	90	97	Weitek FPU; 64 bits
Sun Sparc 4/110	Sparc	—	—	14.30	7.0	69	75	

Table 4.
Summary of Khornerstone benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	Khornerstones/s Low	Khornerstones/s High	Comments
ALR FlexCache 25386	80386	64	5	25.00	—	3,917	3,917	Unix OS
Altos Series 2000	80386	—	—	16.00	3.0	3,038	3,038	Xenix
Apollo Series 10000	Prop. RISC	—	—	18.20	16.0	54,768	54,768	Up to four CPUs
Apollo Series DN3000	68020	—	—	12.50	1.2	2,469	2,469	
Apollo Series DN4000	68020	0	4	25.00	4.0	5,296	5,296	
Compaq 386	80386	0	4	16.00	—	1,941	1,941	
Compaq 386/20	80386	0	4	20.00	—	3,902	4,418	L = MS-DOS, H = Unix
Compaq 386/25	80386	32	5	25.00	—	5,037	7,417	L = SCO Unix
DEC Vaxstation 2000	Proprietary	—	—	5.00	0.9	2,181	2,191	
Decstation 3100	R2000	128	8	16.70	14.0	5,206	5,206	
Dell System 325	80386	32	4	25.00	—	7,001	7,001	Unix
HP 9000 Mod. 340	68030	—	—	16.70	—	4,880	4,880	
HP 9000 Mod. 360	68030	0	4-12	25.00	4.5	6,639	6,995	L = 68882; H = FPA
HP 9000 Mod. 835S	Prop. RISC	128	—	15.00	4.0	22,329	22,329	
IBM PS/2 Mod. 155SX	80386	0	2	16.00	—	2,041	2,041	
IBM PS/2 Mod. 70-121	80386	0	4	20.00	—	6,800	6,800	AIX
IBM PS/2 Mod. 70-A21	80386	64	4	25.00	—	3,967	6,800	L = SCO Unix; H = AIX
IBM PS/2 Mod. 80	80386	—	—	16.00	2.8	2,265	2,265	
IBM RT PC	Prop. RISC	—	—	5.90	4.5	5,459	7,455	
IBM RT PC Mod. L135	Prop. RISC	—	—	7.30	6.0	7,296	7,296	W/FPA
MIPS M/120-5	R2000	128	8	16.70	13.0	18,819	23,218	
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	26,177	26,177	
NCR Tower 32/400	68020	8	4	16.70	1.5	2,231	2,949	L = no cache; H = cache
NCR Tower 2/450	68020	8	4	25.00	—	3,932	3,932	
Solbourne 4/600	Sparc	64	16	16.70	7.0	14,515	14,515	
Sun 3/160C	68020	0	4	16.70	2.0	2,982	3,610	68881 FPU
Sun 3/260	68020	0	8-24	25.00	4.0	5,454	6,767	
Sun 3/50	68020	0	4	15.00	1.5	2,732	2,733	
Sun 3/60	68020	0	4	16.70	2.0	3,966	3,966	
Sun 386i Mod. 150	80386	32	8	14.30	3.5	5,317	5,317	w/80387 FPU
Sun 386i Mod. 1250	80386	32	16	25.00	—	5,317	5,317	
Sun 4/260	Sparc	0	16-32	16.70	10.0	11,440	20,729	L = Weitek FPU
Sun Sparc 4/110	Sparc	—	—	14.30	7.0	7,265	7,265	
Tektronix 4319	68020	0	4	20.00	2.5	4,114	4,114	

loads on a system and therefore does not accurately measure multiuser performance. (See Table 4.)

Linpack. This widely used benchmark provides a relative indication of system performance for engineering and scientific applications. Argonne National Laboratories wrote Linpack, which it also maintains. The routine tests vector performance on individual scientific applications while it stresses cache performance. As a linear-equations package, Linpack emphasizes floating-point addition and multiplication.[6] The results, measured in millions of floating-point operations per second (Mflops), are typically derived from a calculation of a 100 × 100 submatrix of linear equations. Since the benchmark is quite large, it does not completely fit into the cache space of most computers. Consequently, memory speed as well as floating-point processor performance can affect test results. Compilers can easily vectorize the routine, which results in higher performance ratings on those vector processor systems.

Linpack uses a set of general-purpose utilities called Basic Linear Algebra Subroutines (BLASs) to do the actual calculations. The benchmark subroutines can come in two forms: coded or Fortran. Some vendors use hand-coded, assembly-language versions of the library package, which typically yield improved benchmark performance results. The Fortran version comes with a set of standard algebraic subroutines (Fortran libraries). These Fortran BLASs come in two forms: unrolled and rolled. The body of the unrolled form contains an inner loop that is coded with multiple statements, while the rolled version contains a single statement that effectively performs the same operation. As would be expected, the rolled version of the Fortran BLAS yields higher (faster) benchmark ratings. (Table 5 reflects single-precision results.)

Livermore Fortran kernel. This benchmark provides insight into the system performance of scientific applications in both vector and nonvector processor environments. Lawrence Livermore National Laboratories developed the code. Since its work is dominated by large scientific calculations that can be vectorized, the laboratory developed this benchmark to evaluate large supercomputer systems. The benchmark has since migrated to the rest of the computer industry—even to personal computers.

The actual test consists of 24 sections of code taken from applications typically run by the laboratory. These kernels inhabit a larger benchmark driver that runs the routines several times, using different input data each time. The driver also checks on the accuracy and timing of the results and produces a statistical report of the test.[2] The results appear as 24 separate numbers (one for each kernel) as Mflop measurements for three different vector lengths, which total 72 results. Various statistical means (arithmetic, geometric, harmonic) provide insight into general-system performance. An analysis performed by Livermore Labs suggests that each statistical mean corresponds to a level of system vectorization.[2] In terms of vectorization, the harmonic, geometric, and arithmetic means approximate 40, 70, and 90 percent. These three results are often interpreted as

Table 5.
Summary of single-precision Linpack benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	Mflops Low	Mflops High	Comments
Altos Series 2000	80386	—	—	16.00	3.0	0.11	0.11	
Apollo DN660	Proprietary	—	—	—	1.0	0.1	0.1	
Apollo Series DN3000	68020	—	—	12.50	1.2	0.07	0.073	
Apollo Series DN4000	68020	0	4	25.00	4.0	0.14	0.14	
Celerity 1200	Proprietary	—	—	—	2.3	300.0	300.0	
Compaq 386/20 w/80387	80386	0	4	20.00	—	0.14	0.26	
Compaq 386/20 w/Weitek	80386	0	4	20.00	—	0.64	0.64	
Data Gen. MV10000	Proprietary	—	—	—	2.5	390.0	390.0	
DEC VAX 11/785	Proprietary	—	—	7.50	1.5	0.23	0.23	w/FPA
DEC Vaxstation 2000	Proprietary	—	—	5.00	0.9	0.162	0.162	
HP 9000 Mod. 340	68030	—	—	16.70	—	0.167	0.168	
HP 9000 Mod. 360	68030	0	4-12	25.00	4.5	0.24	0.66	L = 68882; H = FPA
HP 9000 Mod. 835SRX	Prop. RISC	128	—	15.00	4.0	2.29	2.29	
IBM PC AT	80286	—	—	—	—	0.013	0.013	
IBM RT PC	Prop. RISC	—	—	5.90	4.5	0.11	0.36	L = 68881; H = FPA
IBM RT PC Mod. 135	Prop. RISC	—	—	7.30	6.0	0.44	0.44	w/FPA
Motorola MVME181-2	88100	32	—	25.00	—	1.8	1.82	
Motorola MVME188SP-1	88100	128	16	20.00	17.0	1.71	1.77	
Motorola SYS8800	88100	128	16	20.00	17.0	1.71	1.77	
NCR Tower 32/400	68020	8	4	16.70	1.5	0.095	0.1	L = no cache; H = cache
NCR Tower 32/450	68020	8	4	25.00	—	0.217	0.217	
Pyramid 90x/FP	Proprietary	—	—	8.00	2.5	200.0	200.0	
Sun 3/50	68020	0	4	15.00	1.5	0.092	0.092	
Sun 386i Mod. 150	80386	32	8	14.30	3.5	0.25	0.25	w/80387 FPU
Tektronix 4319	68020	0	4	20.00	2.5	0.105	0.105	

Table 6.
Summary of single-precision Livermore Fortran kernel benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	Mflops Low	Mflops High	Comments
Acer 1100/25	80386	32	4	25.00	—	0.097	0.12	L = DOS/X; H = Unix
Alliant FX-1, Scalar	Proprietary	—	—	12.00	—	0.66	0.66	
Alliant FX-1, Vector	Proprietary	—	—	12.00	—	0.6	0.6	
Alliant FX/8; MP = 8	Proprietary	—	—	—	35.0	1.3	1.31	
ALR FlexCache 25386	80386	64	5	25.00	—	0.08	0.12	L = DOS/X; H = Unix
AMI Mark II	80386	64	—	33.00		0.3	0.3	w/Weitek 3167
AST 386/33	80386	0	—	33.00	—	0.13	0.275	L = 80387; H = Weitek 3167
AST 386C-390	80386	64	4	25.00	—	0.063	0.08	L = DOS/X; H =Unix
Cheetah cAT 386	80386	0	5	20.00	—	0.05	0.05	
Compaq 386/25 w/80387	80386	32	5	25.00	—	0.09	0.133	L = DOS/X; H = DOS/X
Compaq 386/25	80386	32	5	25.00	—	0.22	0.22	w/Weitek
Convex C-1 Scalar	Proprietary	—	—	—	4.5	1.11	1.11	
Convex C-1 Vector	Proprietary	—	—	—	4.5	1.27	1.27	
DEC VAX 11/780 4.3 BSD	Proprietary	—	—	5.00	1.0	0.18	0.18	
DEC VAX 11/780 VMS	Proprietary	—	—	5.00	1.0	0.28	0.28	
DEC VAX 8700 VMS	Proprietary	—	—	—	6.0	1.26	1.26	
Dell System 310	80386	32	3	20.00	—	0.07	0.09	L = DOS/X; H = Unix
Dell System 325	80386	32	4	25.00	—	0.1	0.12	L = DOS/X; H = Unix
Elxsi 6420	Proprietary	—	16	20.00	—	1.31	1.31	
Everex Step 386/25	80386	64	4	25.00	—	0.09	0.11	L = DOS/X; H = Unix
Fivestar 386/20	80386	64	4	20.00	—	0.073	0.091	L = DOS/X; H = Unix
HP Vectra RS/25C	80386	32	4	25.00	—	0.088	0.109	L = DOS/X; H = DOS/X
Hertz 386/25	80386	64	4	25.00	—	0.1	0.11	L = DOS/X; H = Unix
IBM PS/2 Mod. 70-121	80386	0	4	20.00	—	0.06	0.08	L = DOS/X; H = Unix
IBM PS/2 Mod. 70-A21	80386	64	4	25.00	—	0.09	0.11	L = DOS/X; H = Unix
Micro Express 386/25	80386	64	4	25.00	—	0.101	0.121	L = DOS/X; H = DOS/X
MIPS M/1000	R2000	128	16	15.00	15.0	2.02	2.29	
MIPS M/120-5	R2000	128	8	16.70	13.0	2.58	2.85	
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	3.8	4.24	
MIPS M/500	R2000	24	8	8.00	8.0	0.97	0.97	
MIPS RC2030	R2000	64	16	16.70	12.0	2.82	2.82	
Proteus 4400GL	80386	64	4	25.00	—	0.101	0.121	L = DOS/X; H = DOS/X
Rupp 386/20	80386	0	5	20.00	—	0.056	0.056	
Sun 3/160	68020	0	4	16.70	2.0	0.46	0.46	w/FPA
Sun 3/260	68020	0	8-24	25.00	4.0	0.65	1.04	
Sun 3/60	68020	0	4	16.70	2.0	0.14	0.14	
Sun 386i Mod. 250	80386	32	16	25.00	—	0.08	0.08	
Sun 4/260	Sparc	0	16-32	16.70	10.0	1.03	1.04	w/Weitek
Sun Sparc 4/110	Sparc	—	—	14.30	7.0	0.77	0.77	
Tandon 386/20	80386	64	4	20.00	—	0.08	0.09	L = DOS/X; H = Unix
Tandy 4000 LX	80386	0	16	20.00	—	0.07	0.08	L = DOS/X; H = Unix

separate benchmark tests. (Table 6 reflects single-precision results at 40-percent vectorization.)

SPICE. The general-purpose Simulation Program with Integrated Circuit Emphasis came from the University of California at Berkeley. This benchmark makes heavy use of both integer and double-precision, floating-point calculations (the floating-point operations are not vector oriented). Because it is quite large, the program is a good test of system instruction and data-cache performance. SPICE accepts a circuit description as input and simulates the design. The user can monitor currents and voltages at various circuit locations.

Table 7.
Summary of SPICE benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	Seconds Low	Seconds High	Comments
Apollo Series 10000	Prop. RISC	—	—	18.20	16.0	5.0	5.1	Up to four CPUs
DEC VAX 11/780	Proprietary	—	—	5.00	1.0	154.4	154.4	w/FPA; Unix 4.2 BSD
DEC VAX 8600	Proprietary	—	—	—	4.5	28.0	28.0	Unix
Decstation 3100	R2000	128	8	16.70	14.0	32.13	32.13	
Sun 3/260	68020	0	8-24	25.00	4.0	31.0	31.0	
Sun 4/260	Sparc	0	16-32	16.70	10.0	19.0	19.0	
Sun 4/260	Sparc	0	16-32	16.70	10.0	61.78	61.78	w/Weitek
Sun Sparc 4/110	Sparc	—	—	14.30	7.0	79.87	79.87	

UC Berkeley and several system vendors have distributed various input data packs for simulation of different types of circuits. The user must take care to ensure that benchmark results from different systems have used the same circuit simulations. The number of seconds required to perform the simulation constitutes the test-result measurements. (Table 7 reflects data from 2G6 circuit simulations only.)

Stanford Integer. John Hennessey of Stanford University compiled this benchmark test. Written in the C programming language, the suite consists of a series of small programs that use algorithms to solve real-world problems.

Some of these small programs include the Towers of Hanoi and the Eight Queens puzzles, multiplication of integer matrices, and the quick and bubble sorts. Yet another routine inserts and recursively searches a binary tree. Test measurements consist of the geometric mean of all results displayed in either seconds or milliseconds (See Table 8.)

Stanford Floating Point. The program uses tight loops and a large proportion of floating-point code to calculate the results.[2] The routines' susceptibility to code optimization by high-quality compilers affects the results. The suite consists of the fast Fourier transform (FFT) and matrix multiplication (MM) tests. The first test typically computes a 256-point, single-precision FFT 20 times. The second test multiplies two 40×40 single-precision matrices. The results appear in either seconds or milliseconds. (See Table 8.)

Table 8.
Summary of Stanford benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	Milliseconds Low	Milliseconds High	Comments
Stanford Integer								
DEC VAX 11/780	Proprietary	—	—	5.00	1.0	1,550	2,140	
DEC VAX 8550	Proprietary	—	—	—	6.4	260	350	
DEC VAX 8600	Proprietary	—	—	—	4.5	620	860	
DEC VAX 8810	Proprietary	—	—	22.22	12.0	6.5	6.5	
Decstation 3100	R2000	128	8	16.70	14.0	115	115	
MIPS M/1000	R2000	128	16	15.00	15.0	130	130	
MIPS M/120-5	R2000	128	8	16.70	13.0	112	118	
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	67	75	
MIPS M/500	R2000	24	8	8.00	8.0	260	260	
MIPS M/800	R2000	128	8	15.00	15.0	160	160	
MIPS RC2030	R2000	64	16	16.70	12.0	110	110	
Sun 3/160	68020	0	4	16.70	2.0	530	530	

Table 8.
Summary of Stanford benchmark test results (continued).

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	Milliseconds Low	Milliseconds High	Comments
Sun 3/260	68020	0	8-24	25.00	4.0	360	360	
Sun 4/260	Sparc	0	16-32	16.70	10.0	150		150 w/Weitek
Sun Sparc 4/110	Sparc	—	—	14.30	7.0	222	222	
Stanford Floating Point								
DEC μVAX II w/Ultrix	Proprietary	—	—	5.00	0.9	3,000	3,000	Single-precision MM
DEC μVAX II w/Ultrix	Proprietary	—	—	5.00	0.9	4,900	4,900	256-point FFT
DEC μVAX II w/VMS	Proprietary	—	—	5.00	0.9	1,900	1,900	Single-precision MM
DEC μVAX II w/VMS	Proprietary	—	—	5.00	0.9	3,300	3,300	256-point FFT
DEC VAX 11/780	Proprietary	—	—	5.00	1.0	2,040	2,310	Single-precision MM
DEC VAX 11/780	Proprietary	—	—	5.00	1.0	3,870	3,970	256-point FFT
DEC VAX 8600	Proprietary	—	—	—	4.5	1,370	1,370	Single-precision MM
DEC VAX 8600	Proprietary	—	—	—	4.5	1,420	1,420	256-point FFT
MIPS M/1000	R2000	128	16	15.00	15.0	120	120	Single-precision MM
MIPS M/1000	R2000	128	16	15.00	15.0	170	170	256-point FFT
MIPS M/120-5	R2000	128	8	16.70	13.0	54	70	Single-precision MM
MIPS M/120-5	R2000	128	8	16.70	13.0	95	100	256-point FFT
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	35	43	Single-precision MM
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	64	64	256-point FFT
MIPS M/500	R2000	24	8	8.00	8.0	260	260	Single-precision MM
MIPS M/500	R2000	24	8	8.00	8.0	340	340	256-point FFT
MIPS M/800	R2000	128	8	15.00	15.0	120	130	Single-precision MM
MIPS M/800	R2000	128	8	15.00	15.0	170	200	256-point FFT
MIPS RC2030	R2000	64	16	16.70	12.0	55	55	Single-precision MM
MIPS RC2030	R2000	64	16	16.70	12.0	98	98	256-point FFT
Sun 3/160	68020	0	4	16.70	2.0	498	500	Single-precision MM
Sun 3/160	68020	0	4	16.70	2.0	840	840	256-point FFT
Sun 3/260	68020	0	8-24	25.00	4.0	250	380	Single-precision MM
Sun 3/260	68020	0	8-24	25.00	4.0	460	600	256-point FFT
Sun 3/60	68020	0	4	16.70	2.0	870	870	Single-precision MM
Sun 3/60	68020	0	4	16.70	2.0	1,550	1,550	256-point FFT
Sun 4/260 w/Weitek	Sparc	0	16-32	16.70	10.0	250	263	Single-precision MM
Sun 4/260 w/Weitek	Sparc	0	16-32	16.70	10.0	460	560	256-point FFT

Whetstone. This benchmark is a synthetic mix of integer and floating-point calculations, transcendental functions, conditional jumps, function calls, and array indexing. The code usually comes in a Fortran version. Considered a classic, the benchmark was designed to represent typical scientific programs.

The original version emerged from an analysis of 949 Algol-60 programs.[7] The structure of Whetstone defies vectorization by many optimizing compilers. Results display in thousands or millions of Whetstone interpreter instructions per second (KWhips or MWhips, sometimes referred to as MegaWhetstones/s).

This benchmark has fallen out of favor with the computer industry in recent years. So many different versions of Whetstone (written in either Fortran or the C programming language) make correlating the data derived from two different sources almost impossible. (Table 9 reflects single-precision results.)

Table 9.
Summary of single-precision Whetstone benchmark test results.

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	MWhips Low	MWhips High	Comments
Acer 1100/25	80386	32	4	25.00	—	2.05	2.42	L = Unix; H = DOS/X
Alliant FX/8; MP = 8	Proprietary	—	—	—	35.0	4.9	4.9	
ALR FlexCache 25386	80386	64	5	25.00	—	2.03	2.098	
Altos Series 2000	80386	—	—	16.00	3.0	1.18	1.18	
AMI Mark II	80386	64	—	33.00	—	7.11	7.11	w/Weitek 3167
Apollo 5X0T	68020	—	—	20.00	3.4	7.74	2.2	L = no FPU; H = FPU
Apollo DN660	Proprietary	—	—	—	1.0	0.69	0.69	
Apollo Series 10000	Prop. RISC	—	—	18.20	16.0	16.97	16.97	Up to four CPUs
Apollo Series DN3000	68020	—	—	12.50	1.2	0.716	0.78	
Apollo Series DN4000	68020	0	4	25.00	4.0	1.89	2.17	
Apollo Series DN4500	68030	64	16	33.00	7.0	3.23	3.23	
Apple Macintosh II	68020	—	—	15.70	—	0.52	0.05	
Apple Macintosh IIx	68030	0	4	16.70	—	0.73	0.73	
Apple Macintosh IIxc	68030	0	4	16.70	—	0.73	0.73	
Apple Macintosh SE	68020	0	—	—	—	0.05	0.05	
Apple Macintosh SE/30	68030	0	4	16.70	—	0.73	0.73	
AST 386/33	80386	0	—	33.00	—	3.66	7.05	L = 80387; H = Weitek 3167
AST 386C-390	80386	64	4	25.00	—	3.66	7.05	L = OS/2; H = DOS/X
CCI Power 7/64	Proprietary	—	—	—	—	14.07	14.07	
Celerity 1200	Proprietary	—	—	—	2.3	0.23	0.23	
Celerity 1260	Proprietary	—	—	—	6.2	0.88	0.88	
Cheetah cAT 386	80386	0	5	20.00	—	1.11	1.53	L = Unix; H = DOS/X
Compaq 386/20 w/80387	80386	0	4	20.00	—	1.72	1.89	
Compaq 386/20 w/Weitek	80386	0	4	20.00	—	4.12	4.31	
Compaq 386/25 w/80387	80386	32	5	25.00	—	2.0	2.4	L = Unix; H = DOS/X
Compaq 386/25 w/Weitek	80386	32	5	25.00	—	5.22	5.22	
Data Gen. MV10000	Proprietary	—	—	—	2.5	2.8	2.8	
Data Gen. MV1400 DC	Proprietary	0	12	6.25	—	0.97	0.97	
Data Gen. MV15000-10	Proprietary	16	64	11.80	4.0	4.6	4.6	
Data Gen. MV15000-20	Proprietary	16	64	11.80	8.0	7.13	7.13	
Data Gen. MV15000-8	Proprietary	16	64	11.80	4.0	3.06	3.06	
Data Gen. MV2000 DC	Proprietary	0	12	6.25	—	0.9	0.97	
Data Gen. MV20000	Proprietary	16	64	—	10.0	5.9	5.9	
Data Gen. MV2500 DC	Proprietary	0	24	20.00	—	1.62	1.62	
Data Gen. MV7800	Proprietary	0	14	3.13	—	1.13	1.13	
Data Gen. MV7800 DC	Proprietary	0	14	3.13	—	1.13	1.13	
Data Gen. MV7800 DCX	Proprietary	0	14	4.50	—	1.58	1.58	
Data Gen. MV7800 XP	Proprietary	0	14	4.50	—	1.58	1.58	
DEC µVAX 3500 w/VMS	Proprietary	—	—	11.10	3.5	0.47	0.47	
DEC µVAX II w/Ultrix	Proprietary	—	—	5.00	0.9	0.4	0.4	
DEC µVAX II w/VMS	Proprietary	—	—	5.00	0.9	0.93	0.93	
DEC VAX 11/780	Proprietary	—	—	5.00	1.0	1.313	1.313	
DEC VAX 11/780 4.3 BSD	Proprietary	—	—	5.00	1.0	0.5	1.08	
DEC VAX 11/785	Proprietary	—	—	7.50	1.5	1.8	1.8	w/FPA
DEC VAX 8600	Proprietary	—	—	—	4.5	4.6	4.6	
DEC VAX 8650	Proprietary	—	—	—	6.2	6.1	6.9	
DEC VAX 8700	Proprietary	—	—	—	6.0	5.9	6.67	
DEC Vaxstation 2000	Proprietary	—	—	5.00	0.9	0.47	0.47	
Decstation 3100	R2000	128	8	16.70	14.0	11.5	13.0	
Definicon Sparc 1	Sparc	—	4	20.00	—	4.4	4.4	
Dell System 310	80386	32	3	20.00	—	1.6	2.0	L = Unix; H = DOS/X
Dell System 325	80386	32	4	25.00	—	2.0	2.4	L = Unix; H = DOS/X
Everex Step 386/25	80386	64	4	25.00	—	2.0	2.4	L = Unix; H = DOS/X
Fivestar 386/20	80386	64	4	20.00	—	1.54	1.83	L = Unix; H = DOS/X
HP 9000 Mod. 340	68030	—	—	16.70	—	1.713	1.73	
HP 9000 Mod. 360	68030	0	4-12	25.00	4.5	2.1	3.0	L = 68882; H = FPA

Table 9.
Summary of single-precision Whetstone benchmark results (continued).

System/model	Processor	Cache (Kbytes)	RAM (Mbytes)	Frequency (MHz)	MIPS rating	MWhips Low	MWhips High	Comments
HP 9000 Mod. 360	68030	0	4-12	25.00	4.5	2.1	3.0	L = 68882; H = FPA
HP 9000 Mod. 370	68030	64	8-48	33.00	7.0	3.41	3.41	
HP 9000 Mod. 825S	Prop. RISC	16	—	12.50	3.0	3.5	3.58	
HP 9000 Mod. 835S	Prop. RISC	128	—	15.00	4.0	9.0	9.33	
HP 9000 Mod. 840S	Prop. RISC	128	24	8.00	4.5	3.1	3.1	
HP 9000 Mod. 850S	Prop. RISC	—	—	13.70	7.0	4.2	8.1	
HP Vectra RS/25C	80386	32	4	25.00	—	1.95	2.3	L = Unix; H = DOS/X
Harris HCX-7	Proprietary	—	—	—	7.7	7.1	7.1	
Hertz 386/25	80386	64	4	25.00	—	2.0	2.3	L = Unix; H = DOS/X
IBM 3081D	—	—	—	—	—	5.85	5.85	
IBM 3090	—	—	—	—	10.0	18.0	18.0	
IBM PS/2 Mod. 70-121	80386	0	4	20.00	—	1.35	1.63	L = Unix; H = DOS/X
IBM PS/2 Mod. 70-A21	80386	64	4	25.00	—	1.94	2.538	L = Unix; H = AIX
IBM RT PC	Prop. RISC	—	—	5.90	4.5	0.81	1.64	L = 68881; H = FPA
IBM RT PC Mod. 135	Prop. RISC	—	—	7.30	6.0	2.12	2.12	w/FPA
Integr. Sol. Advantage2000	R2000	64	16	16.70	12.0	11.4	11.4	
Intergraph Interpro 32C	Clipper RISC	—	—	30.00	5.0	2.98	2.98	
Micro Express 386/25	80386	64	4	25.00	—	2.03	2.44	L = Unix; H = DOS/X
MIPS M/1000	R2000	128	16	15.00	15.0	10.28	10.5	
MIPS M/120-3	R2000	128	8	12.50	10.0	9.1	9.1	
MIPS M/120-5	R2000	128	8	16.70	13.0	11.4	12.1	
MIPS M/2000	R3000	128	32	20/25.00	16-20.0	13.8	18.0	L = 20 MHz; H = 25 MHz
MIPS M/500	R2000	24	8	8.00	8.0	5.43	5.43	R2360 FPU
MIPS M/800	R2000	128	8	15.00	15.0	8.57	8.57	
MIPS RC2030	R2000	64	16	16.70	12.0	12.1	12.1	
Motorola MVME141-1	68030	64	4	25.00	5.3	0.69	1.88	
Motorola MVME141-2	68030	64	4	33.00	6.9	2.5	2.5	
Motorola MVME147	68030	0	4	20.00	3.8	1.43	1.51	
Motorola MVME147-1	68030	0	4	25.00	4.7	1.79	1.96	
Motorola MVME180	88100	32	—	20.00	17.0	12.7	18.2	
Motorola MVME181-2	88100	32	—	25.00	—	15.9	22.8	
Motorola SYS1147	68030	0	4	20.00	3.8	1.43	1.43	
Motorola SYS2300	68020	0	4	16.70	1.5	0.93	0.93	
Motorola SYS2600	68020	16	4	16.70	1.5	0.87	0.87	
Motorola SYS3300	68030	0	4	20.00	3.8	1.43	1.43	
Motorola SYS3600	68030	0	4	25.00	4.7	1.79	1.79	
Motorola SYS3640	68030	64	4	25.00	5.3	1.79	1.79	
NCR Tower 32/400	68020	8	4	16.70	1.5	0.864	0.933	L = no cache; H = cache
NCR Tower 32/450	68020	8	4	25.00	—	1.65	1.65	
Opus Systems	88000	32	4-20	20.00	17.0	18.58	18.58	
Opus Systems	88000	32	4-20	20.00	17.0	9.43	9.43	
Prime EXL 316	80386	—	—	16.00	3.2	3.3	3.3	
Proteus 4400GL	80386	64	4	25.00	—	2.03	2.44	L = Unix; H = DOS/X
Rupp 386/20	80386	0	5	20.00	—	1.11	1.53	L = Unix; H = DOS/X
Silicon Graphics Iris	R2010	24	8	12.50	—	7.35	7.35	
Solbourne 4/600	Sparc	64	16	16.70	7.0	6.65	6.65	
Sun 3/160C w/68881	68020	0	4	16.70	2.0	1.03	2.6	L = 68881; H = FPA
Sun 3/260	68020	0	8-24	25.00	4.0	1.25	5.3	L = 68881; H = FPA
Sun 3/50	68020	0	4	15.00	1.5	0.71	0.936	w/68881
Sun 3/60	68020	0	4	16.70	2.0	1.27	1.42	
Sun 386i Mod. 150	80386	32	8	14.30	3.5	1.92	1.92	w/80387 FPU
Sun 386i Mod. 250	80386	32	16	25.00	—	1.63	1.923	
Sun 4/260 w/Weitek	Sparc	0	16-32	16.70	10.0	4.6	5.66	Weitek FPU
Sun Sparc 4/110	Sparc	—	—	14.30	7.0	4.22	4.28	
Tandon 386/20	80386	64	4	20.00	—	1.5	1.9	L = Unix; H = DOS/X
Tandy 4000 LX	80386	0	16	20.00	—	1.43	1.71	L = Unix; H = DOS/X

The list of public-domain, licensed third-party, and proprietary benchmark tests in use today seems endless. This plethora, combined with the various benchmark revisions, testing procedures, and analysis methods, complicates the task of comparing general system performance. Understanding simple concepts like what a benchmark does—and how the results are produced—provides a valuable first step in the right direction.

Acknowledgment

I especially thank Tom Ritch of the Motorola Microcomputer Division for editorial and technical assistance during the development of this article.

References

1. Jim Geers, "A New Generation of Benchmarking," *Mips*, Vol. 1, No. 1, Feb. 1989, pp. 92-100.
2. *Performance Brief—CPU Benchmarks*, Issue 3.5, Mips Computer Systems, Sunnyvale, Calif., Oct. 1988.
3. *Am29000 Performance Analysis*, Advanced Micro Devices, Sunnyvale, Calif., May 1988.
4. David Bouffard et al., *DECstation 3100 Performance Summary*, Version 1.0, Digital Equipment Corporation, Maynard, Mass., Jan. 10, 1989.
5. David Wilson, "Tested Mettle," *UNIX Review*, Volume 7, No. 1, Jan. 1989, pp. 97-107.
6. J.J. Dongarram, *Performance of Various Computers Using Standard Linear Equations Software in a Fortran Environment*, Argonne National Laboratory, Argonne, Ill., Feb. 16, 1988.
7. Bill Nicholls, "The 'B' Word!," *Byte*, June 1988, pp. 207-212.

Walter J. Price is a senior marketing analyst with the Microcomputer Division of Motorola, where he performs competitive and market analyses of board- and system-level products. His current interests include collecting comprehensive information regarding the microcomputer industry.

Price received the BS degree in electronics technology and computer science from Northern Michigan University in Marquette.

Questions about this article may be directed to the author at Motorola, Microcomputer Division, 2900 South Diablo Way, Tempe, AZ 85282.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Interest Card.

Low 165 **Medium** 166 **High** 167