Reed Electronics Group

MICROPROCESSOR REPORT

www.MPRonline.com

THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

EEMBC's Dhrystone Killer

Free CoreMark Benchmark Aims to Retire Dhrystone Forever

By Tom R. Halfhill {6/8/09-01}

Markus Levy thinks Dhrystone is all wet. Levy has been on the warpath against the hoary Dhrystone benchmark since founding the Embedded Microprocessor Benchmark Consortium (EEMBC) in 1997. EEMBC has spent nine years introducing several suites of respected benchmarking programs—and, perhaps more important, empirical testing and verification procedures to go with them. Nevertheless, Dhrystone remains the most widely quoted measure of microprocessor performance.

On June 1, EEMBC introduced an alternative: CoreMark. It's an all-new benchmarking program that anyone can download and use. CoreMark isn't a substitute for the EEMBC suites, which remain a more sophisticated and comprehensive way of measuring performance. But it's free, portable, easy to use, and produces an easy-to-understand score.

CoreMark has all the advantages of Dhrystone (except for entrenched longevity) while modernizing the workload and eliminating some of Dhrystone's worst flaws. For the sake of the microprocessor industry, Levy hopes it will be the ultimate Dhrystone killer. (*Editor's note: Levy is a member of the* Microprocessor Report *editorial board and a former* MPR *analyst.*)

A Benchmark for Archaeologists

Why is Dhrystone so hard to budge? Its best quality is that it's free. Anyone can download and run it. With the exception of EEMBC's GrinderBench for cellphones, the regular EEMBC benchmark suites are available only to consortium members and licensees. In addition, Dhrystone is processor-agnostic, having been ported to virtually every CPU architecture on the planet. And the source code is open, so anyone can recompile it for new architectures.

Dhrystone is easy to use, unencumbered by the formal testing and verification requirements that make EEMBC's benchmarks more rigorous. (See *MPR 5/1/00-02*, "EEMBC Releases First Benchmarks," and *MPR 6/21/99-01*, "Embedded Benchmarks Grow Up.") Dhrystone produces a single figure of merit—Dhrystone millions of instructions per second (mips), or Dmips—that's easy to grasp and leaves little room for interpretation. Dhrystone is such a small program that it doesn't stress the memory system or I/O channels, so it focuses exclusively on the performance of the microprocessor core.

Despite its advantages and popularity, Dhrystone is also the most widely disparaged benchmark. For one thing, it's positively ancient, by computer-industry standards. Originally written in Ada by Reinhold Weicker in 1984, it was ported to C for Unix a few months later, revised in 1988, and finalized that same year. So the "latest" version, Dhrystone 2.1, has been unchanged for 21 years—a technological fossil. (Weicker named Dhrystone as a pun on Whetstone, an unrelated floating-point benchmark developed in 1972.)

Dhrystone executes simple loops of integer-only workloads that poorly reflect today's software. Lacking a formally defined testing methodology, it's easily manipulated. It's easy prey for modern compilers and subversive testers, who can optimize some tasks completely out of existence. It fits entirely within the L1 cache of almost all processors that have a cache, simulating an almost impossibly perfect memory system.

To derive a single figure of merit, most testers divide the raw Dhrystone score by 1,757. That formula expresses Dmips as a multiple of a DEC VAX 11/780—a 1.0Dmips minicomputer from 1977, now found in museums. But the Dmips formula isn't standardized. Some testers use different divisors, indexing the Dmips score to variants of the VAX 11/780.

EEMBC's regular benchmark suites, testing methodology, and certification rules address all of Dhrystone's shortcomings. In addition, the EEMBC suites focus on specific application areas that are much more relevant to modern embedded computing. (See *MPR 2/22/05-01*, "EEMBC Expands Benchmarks.")

Nevertheless, Dhrystone lingers on, mainly because joining EEMBC to use the suites costs money. A full commercial membership costs several thousand dollars. (Universities and some companies can license an EEMBC benchmark suite for as little as $150 without joining the consortium; EEMBC has more than 100 licensees.) If CoreMark succeeds, it will finally turn Dhrystone into an artifact as historical as the hardware for which it was developed.

Small But Fiesty

Like all EEMBC benchmark software, CoreMark is a collaborative project based on input from the nonprofit consortium's 50-plus member companies and organizations. Agreeing on the form, goals, and composition of any benchmark can take longer than writing the program code. CoreMark was no exception. It provoked debate among consortium members, as we will discuss below.

Usually, EEMBC members collaborate on the code, too. In this case, because CoreMark is a single, small program, all the code was written by Shay Gal-On, EEMBC's director of software engineering. The whole project, including committee work, took about eight months. The C source code is available at a new website (*www.coremark.org*) after free registration.

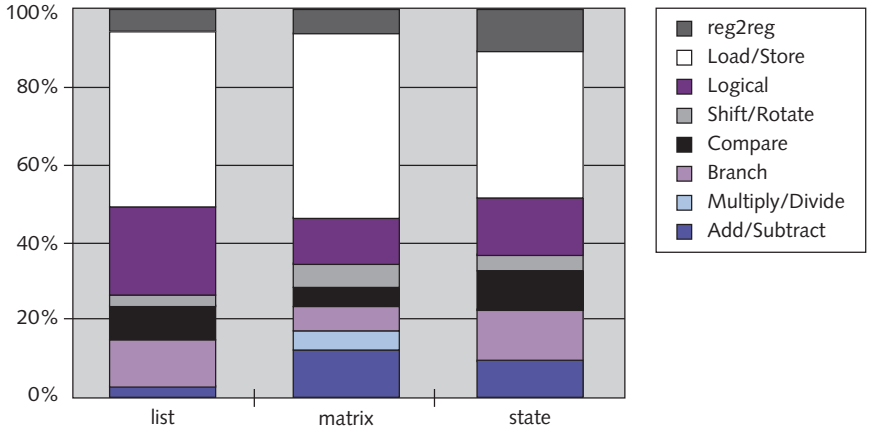

Figure 1. CoreMark instruction profile (Power Architecture). This chart counts the number of each type of instruction in each part of the benchmark program, as compiled for a Power processor. Load/store instructions dominate the code, followed by logical instructions, compares, and branches. Notice that multiply and divide instructions appear only in the matrix-manipulation test. (Data sources: IBM and EEMBC.)

CoreMark shares some characteristics with Dhrystone. It measures integer performance only—no floating point. EEMBC made this decision because CoreMark is intended primarily for embedded processors, including small 8-, 16-, and 32-bit processors still lacking such modern conveniences as FPUs. CoreMark is small, though not quite as small as Dhrystone. Requiring only 2KB of memory at run time, it fits entirely into a small L1 cache, but it's also suitable for benchmarking cacheless cores. In comparison, Dhrystone 2.1 needs only 64 *bytes* of memory at run time.

CoreMark has only one workload and three algorithms—or perhaps four, depending on how they're counted. The first algorithm tests matrix manipulation, using 16-bit integer inputs that generate 16- or 32-bit results, depending on the processor for which CoreMark has been compiled. Processors with efficient multiply-accumulate (MAC) instructions should do well in this test.

The second algorithm tests state-machine operation, or control code. These are mostly byte-size instructions that compare values and branch to other instructions. Some embedded programs are filled with control code, whereas data-intensive programs have relatively little. If the target processor has branch prediction, it will do better in this test.

A third test manipulates a linked list of pointers using outputs from the first two algorithms. Pointers may be 16-, 32-, or 64 bits long, depending on the processor for which CoreMark was compiled. All three tests also exercise basic read/write operations to memory. The bar chart in Figure 1 shows the distribution of instruction types in these three tests.

Figure 2 shows the distribution of instructions in a binary file compiled for the x86 architecture. This chart resembles the Power Architecture profile in Figure 1, but testers had to use different profiling tools for the two architectures, so the instruction labels differ.

Benchmark, Verify Thyself

The final CoreMark test is a cyclic redundancy check (CRC), which serves two purposes. First, CRCs are common in embedded programs, which often use checksums to verify the data integrity of I/O transfers and other operations. Second, CoreMark uses CRCs to check itself—to verify the data integrity of its input and output. This safeguard helps prevent some forms of cheating that would artificially reduce or eliminate the workload.

Another verification feature is that seed values for CoreMark algorithms cannot be determined at compile time. CoreMark generates the seeds at run time and verifies them during the program run. At the end of the run, CoreMark performs a final CRC, reports whether all the tests passed verification, summarizes

information about the test run, and reports the score. The score is the number of iterations per second during the run, so higher scores are better. Figure 3 shows the results of running CoreMark on an x86 processor.

To produce an official score, EEMBC requires CoreMark to run on the target processor for at least 10 seconds. Before compiling the source code, testers can adjust the number of iterations to ensure a test run meets this rule. Regardless of the number of iterations, CoreMark always calculates the final score as iterations per second. Therefore, the score is directly comparable with CoreMark scores for other processors, no matter what their CPU architecture, clock frequency, or bus speed is.

CoreMark scores aren't fractions or multiples of a baseline machine, in the way that Dhrystone-mips scores relate to the VAX 11/780 from 1977. *MPR* thinks EEMBC missed a golden opportunity to define a "CoreMark mips" score indexed to a similarly beloved but more recent baseline computer. Too bad there isn't an embedded system as iconic as the VAX. As a substitute, our tongue-in-cheek suggestion is the best-selling personal computer in history—the Commodore 64.

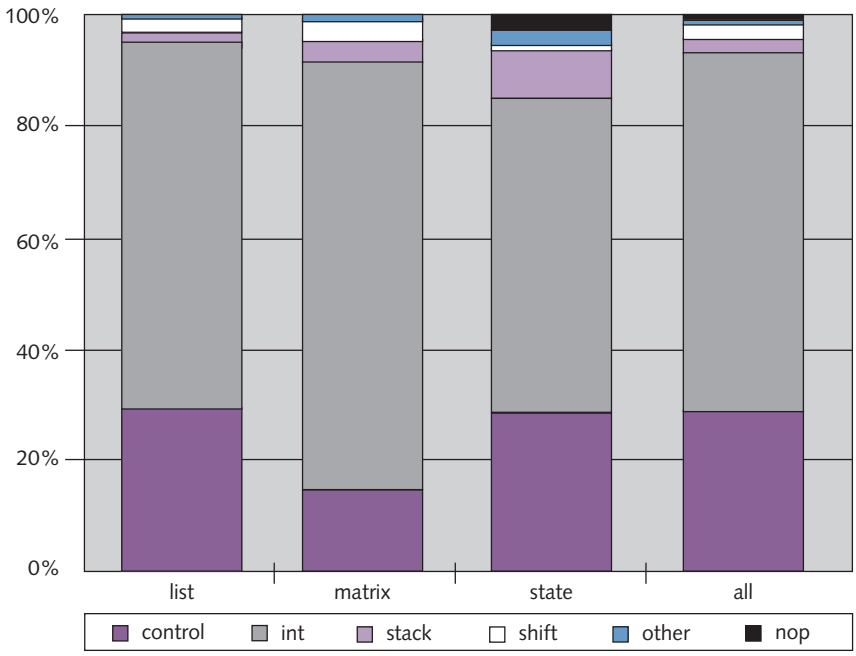

Figure 2. CoreMark instruction profile (x86). Integer operations and control-type instructions (such as compares and branches) dominate the x86 binary. The no-operations (NOP) in the state-machine test represent load-use delays, which the profiling tool used for Figure 1 didn't identify. Because the x86 is a CISC architecture, many instructions combine logical or arithmetic operations with load/store operations, making the instruction profile less precise. (Data source: EEMBC.)

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\halfhillt>d:

D:\>cd m23_coremark

D:\M23_CoreMark>coremark
2K total performance run parameters for coremark.
CoreMark Size    : 666
Total ticks      : 123906
Total time (secs): 12.390600
Iterations/Sec   : 3228.253676
Iterations       : 40000
Compiler version : CL15
Compiler flags   : -O2
Memory location  : CACHE
seedcrc          : 0xe9f5
[0]crclist       : 0x08d8
[0]crcmatrix     : 0x9b05
[0]crcstate      : 0x5882
[0]crcfinal      : 0x9055
Correct operation validated
CoreMark   : 3228.254

D:\M23_CoreMark>_
```

Figure 3. Typical CoreMark results. *MPR* ran this test on a 1.2GHz Intel Core 2 Duo processor with a 533MHz front-side bus. "CoreMark Size" is the buffer for each of the three benchmark tests (one-third of 2KB, or 666 bytes). This processor needed 12.39 seconds to run 40,000 iterations, yielding a score of 3,228.25 iterations per second. Compiler version "CL15" is Microsoft's Visual C++ compiler. The binary code fit entirely in the processor's L1 cache. The remaining output shows that CoreMark verified the input seed and all the output with a cyclic redundancy check (CRC).

Some Optimizations Allowed

As with other EEMBC benchmarks, EEMBC requires CoreMark testers to disclose the development tools used to compile the source code, the optimization flags set for the compiler, and the speed of the memory system attached to the CPU. Of course, EEMBC can't police these rules for everyone, but EEMBC requires these disclosures from testers who post scores on the CoreMark website. Anyone should be able to duplicate the posted results by using the same CPU, memory system, compiler, and flags.

A novel feature of other EEMBC benchmarks is a rule allowing testers to optimize the benchmark code in ways that might be considered cheating if applied to other benchmarking programs. Unmodified EEMBC code produces baseline or "out-of-the-box" scores. Modified code produces optimized or "full fury" scores. These options are a nod to reality. In real-world applications, embedded-software developers often use some rather athletic optimizations.

For example, testers can use any compiler flags they want and even rewrite critical

Price & Availability

EEMBC's CoreMark benchmarking program is free and available now. To download the C source code, register at www.coremark.org. A small part of the program requires porting to the target CPU architecture. In the future, CPU vendors may provide this porting layer (as well as suggested compiler flags) on the CoreMark website.

portions of EEMBC's benchmark kernels in assembly language. As long as testers disclose the compiler flags and the use of any assembly routines, the optimizations are legal under EEMBC's full-fury rules.

EEMBC says CoreMark will continue this tradition, including assembly-language optimizations. An even more intriguing possibility is that testers could modify and compile CoreMark to run some operations in parallel on a multicore processor. CoreMark could be the poor man's alternative to the EEMBC MultiBench suite, which is available only to consortium members and licensees. (See *MPR 7/28/08-01*, "EEMBC's MultiBench Arrives.")

At this point, however, EEMBC hasn't finalized the rules for full-fury CoreMark benchmarking. Until then, CoreMark scores will be considered out-of-the-box scores. EEMBC permits compiler-flag optimizations for out-of-the-box scores as long as testers disclose the flags and use a publicly available compiler.

EEMBC's Seal of Approval

Benchmarking is the most controversial subject in computing. (See *MPR 8/30/04-01*, "Benchmarking the Benchmarks.") To head off some disputes, EEMBC will offer certification services for CoreMark scores, as it does for other EEMBC benchmark scores.

During the certification process, EEMBC will check the compiled code and try to replicate the original tester's reported results. If anything strange turns up, EEMBC will consult with the original testers to resolve the anomalies. Certified scores will be specially noted on the CoreMark website. Certification is free for EEMBC members. At this time, there's no certification option for nonmembers. (It's subtle encouragement to join EEMBC.)

In a radical departure from other EEMBC benchmarks, anyone can use CoreMark to test microprocessors from any vendor and post scores on the CoreMark website. This rule change provoked debate among EEMBC members. For years, members have resisted proposals to allow competitive or independent public benchmarking. Testers must join EEMBC or buy a license to get the benchmark suites, and the consortium's rules forbid members and licensees from publishing scores for any processors but their own. Privately, EEMBC members often use the suites to compare the performance of their processors with those of competitors. Competitive benchmarking is allowed, as long as scores are kept private.

In contrast, CoreMark is wide open. Vendor A can test processors from Vendor B and post scores. Independent parties can test processors from any vendors and post scores. Testers must register on the CoreMark website but need not disclose their identities. EEMBC members may contest the scores by posting their own scores and comments on the site. Indeed, the site is devoting a blog to this purpose.

Nonmembers must suffer in silence—on the CoreMark site, at least. Of course, nonmembers may post anything they want on their own websites, but the CoreMark site won't link to their rebuttals. (Again, it's subtle encouragement to join EEMBC.)

EEMBC hopes CPU vendors will post source code for the portable layer of the benchmark program on the CoreMark site for public download. In this way, vendors can ensure that other testers will use a porting layer optimized for the target processor. For some CPUs, testers may have to compile multiple binaries. Although an x86 binary is nearly universal, binaries for other architectures may be compiled specifically for particular development boards. Binaries compiled for microcontrollers may contain code specific to their integrated peripherals, such as UARTs and timers.

Although CoreMark, like Dhrystone, is a standalone program, testers can use it as part of a larger benchmarking regimen. In particular, CoreMark works with EEMBC's EnergyBench, which measures power consumption. However, EnergyBench isn't free. It's available only to EEMBC members and licensees. (See *MPR 7/17/06-02*, "EEMBC Energizes Benchmarking.")

Analyzing CoreMark Scores

CoreMark is a quick-and-dirty benchmark program, so *MPR* decided to give it a quick-and-dirty test. One goal was to determine if CoreMark can distinguish among different microarchitectures of the same CPU architecture. A good CPU benchmarking program should demonstrate superior scores on superior microarchitectures instead of merely scaling linearly with clock frequency. Our second goal was to compare CoreMark with Dhrystone.

We ran CoreMark and Dhrystone on three different Intel x86 processors. (They are PC processors, not the embedded processors for which CoreMark is primarily intended, but PCs were more convenient.) The processors were a 1.0GHz Celeron, 2.0GHz Celeron, and 1.2GHz Core 2 Duo.

It's important to note that although the first two processors are Celerons, they are very different beasts. The 1.0GHz Celeron is based on Intel's P6 microarchitecture, also found in the Pentium II and Pentium III. In contrast, the 2.0GHz Celeron is based on Intel's later Netburst microarchitecture, which first appeared in the Pentium 4. Core 2 Duo is based on Intel's Core microarchitecture and is a more recent design, currently being superseded by Core i7 (Nehalem).

Because Netburst was an improvement over the P6, one would expect its throughput to exceed its advantage in raw

clock speed. In other words, the 2.0GHz Celeron should be *more than* twice as fast as the 1.0GHz Celeron. And indeed, as the scores in Table 1 show, CoreMark found the 2.0GHz Celeron to be three times faster than the 1.0GHz Celeron.

Likewise, CoreMark rates the much newer Core 2 Duo processor more favorably than the Netburst Celeron, despite a significant disparity in clock frequency. Core 2 Duo runs at only 1.2GHz, whereas the Netburst Celeron runs at 2.0GHz—yet CoreMark says Core 2 Duo is 17% faster. This result is what we would expect from a benchmark program that distinguishes between different microarchitectures and isn't fooled by raw clock speed.

	Intel Celeron	Intel Celeron	Intel Core 2 Duo
CPU Architecture	x86	x86	x86
Microarchitecture	P6 (Pentium II / III)	Netburst (Pentium 4)	Core (Core 2)
Processor Cores	1	1	2
Core Frequency (Difference)	1.0GHz —	2.0GHz (+100% vs. 1.0GHz Celeron)	1.2GHz (–40% vs. 2.0GHz Celeron)
Bus Frequency (Difference)	400MHz —	400MHz —	533MHz (+33% vs. Celerons)
Dhrystone 2.1 (Difference)	256 —	280 (+9% vs. 1.0GHz Celeron)	621 (+122% vs. 2.0GHz Celeron)
CoreMark (Difference)	889 —	2748 (+209% vs. 1.0GHz Celeron)	3223 (+17% vs. 2.0GHz Celeron)

Table 1. CoreMark and Dhrystone scores for three Intel x86 processors, as benchmarked by *MPR*. CoreMark reports credibly higher scores for newer microarchitectures instead of simply scaling the scores at a linear rate with the processors' clock speeds. The Dhrystone scores are much less credible. They vary wildly and have little relationship to either clock speed or microarchitecture. (*MPR* submitted these CoreMark scores to EEMBC, and they became the first results posted on the new CoreMark website.)

In contrast, the Dhrystone scores are simply unbelievable. Dhrystone says the 2.0GHz Celeron with newer Netburst design is only 9% faster than the 1.0GHz Celeron with older P6 design. Even if Dhrystone scaled linearly with clock frequency, the 2.0GHz Celeron should be 100% faster.

Dhrystone also reports a wildly different result when benchmarking Core 2 Duo. It says the newer 1.2GHz processor is 122% faster than the 2.0GHz Celeron, whereas CoreMark says Core 2 Duo is only 17% faster. CoreMark is more credible, because Core 2 Duo is running at a much lower clock speed but has an improved microarchitecture.

Although Core 2 Duo has a faster front-side bus than these Celerons (533MHz vs. 400MHz), it's irrelevant for benchmark programs that fit entirely into the L1 caches of these processors. And although Core 2 Duo has twice as many processor cores as these Celerons, our versions of CoreMark and Dhrystone were not modified for multicore execution, so they were running on only one core. Therefore, our quick-and-dirty conclusion is that CoreMark is a much better quick-and-dirty benchmark than Dhrystone.

CoreMark's Pros and Cons

MPR coverage of new processors rarely cites scores from the EEMBC benchmark suites. One reason is that some CPU vendors still don't belong to EEMBC. The main reason is that most EEMBC members won't publicly release their scores. Yet they have no qualms about quoting Dhrystone mips. We hope CoreMark will replace—or, at least, supplement—Dhrystone as the improved quick-and-dirty benchmark for new processors. In simulation, CoreMark can even estimate the performance of a new design or soft core that hasn't achieved first silicon.

CoreMark addresses several shortcomings of Dhrystone: its obsolescence, lack of standardized source code, artificial workloads, practically nonexistent testing methodology, compiler vulnerabilities, and easy cheating.

That said, CoreMark has shortcomings, too. Certainly, one small program with a handful of algorithms cannot fully characterize the performance of a modern microprocessor. Even EEMBC admits that. CoreMark is no substitute for the fully fleshed EEMBC suites.

CoreMark is too small to represent most modern embedded software, has no floating-point math, and reports only a single composite score that prevents any performance analysis of the underlying algorithms. CoreMark's small size fulfills its mission of isolating CPU-core performance, but it simulates a perfect memory system—an unrealistic picture of real-world performance, in most cases.

Although CoreMark is outclassed by the sophisticated EEMBC suites, it has three advantages over them: it's not restricted to EEMBC members and licensees, it's free, and anyone can post competitive scores for processors from different vendors. CoreMark has the potential to move embedded-processor benchmarking from EEMBC's monastery to the masses.

The best thing about CoreMark is that it's not Dhrystone. Or, to look at it another way, it's "Dhrystone 3.0," overhauled and updated for the first time in 21 years. Although CoreMark isn't perfect, it's much better than the most widely quoted and embarrassing measure of microprocessor performance the industry is using now.