Benchmarking in context: Dhrystone

By Richard York, ARM Ltd.

"And of course, the very success of a benchmark program is a danger in that people may tune their compilers and/or hardware to it, and with this action make it less useful."

Reinhold P. Weicker, Siemens AG, April 1989 [1]
Author of the Dhrystone Benchmark

The Dhrystone benchmark was first published in Ada, as long ago as 1984 [2]. Now the C version of Dhrystone is the one mainly used in industry. The original Dhrystone benchmark is still used to measure CPU performance today.

The original intent with Dhrystone was to create a short benchmark program that was representative of system (integer) programming. The Dhrystone code is dominated by simple integer arithmetic, string operations, logic decisions, and memory accesses intended to reflect the CPU activities in most general purpose computing applications. The Dhrystone result is determined by measuring the average time a processor takes to perform many iterations of a single loop containing a fixed sequence of instructions that make up the benchmark. When Dhrystone is referenced, it is usually quoted as 'DMIPS', or Dhrystone MIPS/MHz.

Two important questions should be asked of any benchmarking activity:

- o How accurately does the benchmark predict real-world performance?
- o How reliably can a comparison between competing processors be made?

Dhrystone Fundamentals

Dhrystone has a number of attributes that have led to it being widely used in the past as a measure of CPU performance.

Foremost, Dhrystone is compact, widely available in the public domain, and simple to run. Significantly, there are no lengthy certification processes to go through before citing Dhrystone figures.

Dhrystone compares the performance of the processor under benchmark to that of a reference machine. This is an advantage over quoting 'straight' MIPS numbers since using a reference machine effectively compensates for differences in the richness of competing instruction sets. For example, literal comparison of the 'millions of instructions per second' numbers for a RISC architecture and a CISC architecture is not meaningful.

The industry has adopted the VAX 11/780 as the reference 1 MIP machine. The VAX 11/780 achieves 1757 Dhrystones per second. The Dhrystone figure is calculated by measuring the number of Dhrystones per second for the system, and dividing that by 1757. So "80 MIPS" means "80 Dhrystone VAX MIPS", which means 80 times faster than a VAX 11/780. A DMIPS/MHz rating takes this normalization process one step further, enabling comparison of processor performance at different clock rates.

For all of these reasons, in the past, Dhrystone has been a widely quoted benchmark figure. In theory, Dhrystone should provide a basis for the comparison of processor performances.

However, some of the apparent advantages of Dhrystone are also significant weaknesses of the benchmark. Dhrystone numbers actually reflect the performance of the C compiler and libraries, probably more so than the performance of the processor itself. Also, lack of independent certification means that customers are dependent on processor vendors to quote accurate and meaningful Dhrystone data.

Dhrystone and Real-world Applications

It is unlikely that a single benchmark can be representative of the diversity and complexity in today's applications. As well as satisfying the needs of a growing variety of embedded applications, many applications require that the processor also provides support for a suitable operating system (OS). Modern RISC architectures, such as ARM's new v6 architecture, are optimized for efficient OS support, as well as providing acceleration for DSP, multimedia operations and Java byte code execution.

More recent benchmark initiatives, such as those offered by EEMBC [3], provide a variety of benchmarks designed to test a range of application domains, such as networking and consumer. They have been developed in close cooperation with both the microprocessor vendors, and more importantly the embedded system developers who use them.

Dhrystone code is very compact, being of the order of around 100 high-level language statements and occupying just 1-1.5kB of compiled code. Because of its small size, memory access beyond the cache is not exercised. Effectively, Dhrystone is simply testing the performance of the integer core. In the vast majority of today's applications, processor cores are embedded with cache memories. The overall memory hierarchy, and the way that the memory is managed heavily affect system design and performance. For example, enhancements to the ARMv6 architecture memory management system have boosted overall system performance by much as 30% for applications running complex user-interface driven operating systems. Dhrystone will clearly not measure such improvements.

Dhrystone contains two statements where the programming language and its translation play a major part in the benchmark execution time.

o String assignment (in procedure Proc_0 / main) o String comparison (in function Func_2)

The C version of Dhrystone spends a relatively large amount of time in these two functions. Measurements made on a VAX 11/785 with the Berkeley UNIX (4.2) compilers demonstrated that 23% of the time was spent in the string functions.

On good RISC machines (where less time is spent in the procedure calling sequence than on a VAX) and with better optimizing compilers, the percentage is higher. In fact, the ARM9E measurements presented below demonstrate that between 21% and 65% is spent processing these specific string functions. As this data reveals, Dhrystone is not a particularly representative sample of the kinds of instruction sequences that are typical of today's applications. The majority of embedded applications make little use of the C libraries for example, and even desktop applications are unlikely to have such a high weighting of a very small number of specific library calls.

'Optimizing' Dhrystone for Competitive Purposes

As previously stated, a quoted DMIPS figure is susceptible to many other factors, besides the true performance of the CPU. Often, the variation in performance is due to the efficiency of the C compiler, or one of the other factors described below.

Compiler Efficiency

The key issue with Dhrystone is its dependency on the compiler and library technology used – much more so than any other benchmark. ARM has measured significant differences in Dhrystone results by using alternative compilers targeting the same architecture – with performance varying by as much as 100%.

Dhrystone Version

Version 1 of the Dhrystone Benchmark is no longer recommended, since compilers can now eliminate too much 'dead code' from the benchmark. Dhrystone version 2.1 is the most recent release of the benchmark, and the version that ARM uses in benchmarking Dhrystone.

ANSI C

Instead of using the normal Kernighan & Ritchie (K&R) format, the Dhrystone source code is sometimes altered to obey ANSI format and prototypes. This change can give compilers more information, so that better optimization is possible.

Combining Source Files

Combining the two Dhrystone source files into one module enables the compiler to perform additional optimizations on all functions simultaneously. Dhrystone version 1.1 contains the benchmark in a single source file – other than that it is very similar to Dhrystone 2.1.

Code Inlining

Enabling inlining of procedures can enhance performance. The availability of ANSI C has created a new situation, compared with the older K&R C. In the original C, the definition of the string function was not part of the language. Now it is, and inlining of C library functions is explicitly allowed. The Dhrystone rule "No procedure inlining for Dhrystone" refers to the user-level procedures only, and not to the library routines.

Adapting the C Library

Adapting the C library to accelerate Dhrystone code can drastically alter the benchmark results, but this can negatively impact the performance of typical code.

Measuring Dhrystone performance for the same processor and compiler yields a spread of around three times between the 'unoptimized' case and a case where several optimizations are applied together (Figure 1). This test demonstrates how important it is to be extremely careful when interpreting standalone Dhrystone DMIPS figures.

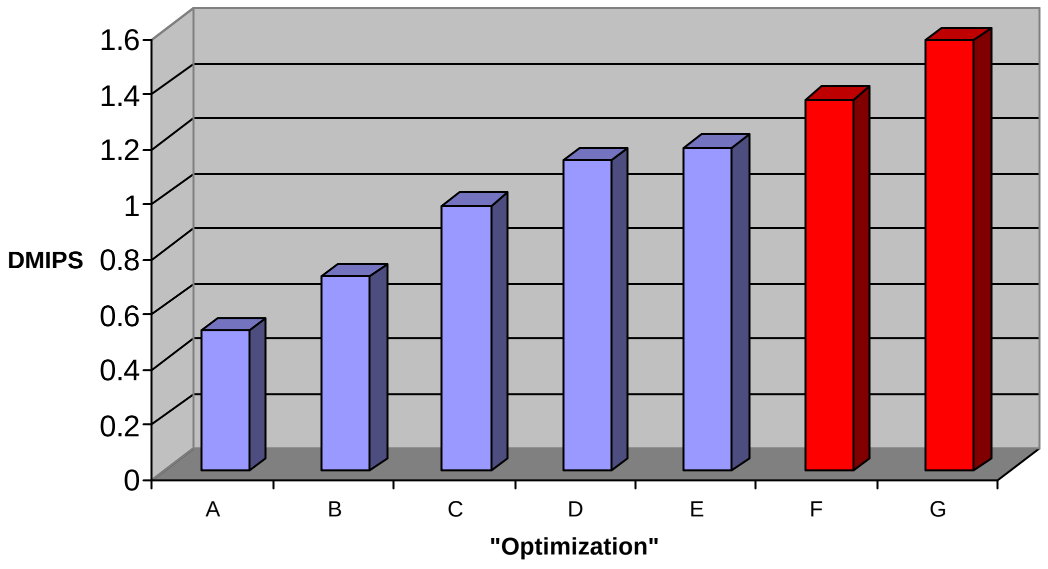

Figure 1. Affecting Dhrystone Performance through Optimization

Benchmark optimization conditions:

ARM C compiler options: -cpu ARM9E –Otime –fy –Onoinline – DMSC_CLOCK

- **A:** No memcpy, strcpy or strcmp optimization
- **B:** Optimized memcpy
- **C:** Optimized strcmp and strcpy
- **D:** Default
- **E:** Optimized division and strcmp

- **F:** Automatic inlining of user routines enabled (-Oinline) (not permitted)
- **G:** File merging, "ANSI-fied" headers, similar to Dhrystone 1.1 (not permitted)

Benchmarks A to C have inlining of the string and memory functions disabled, but instead use an optimized byte-oriented version of the routine. Benchmark D is the default benchmark with ADS1.2 (ARM Development System), and includes inlining of memcpy, strcpy and an optimized strcmp routine. Benchmark E has the C-library functions for division and strcmp optimized for speed at the expense of code size. The resulting C-library is slightly faster on many benchmarks – not just Dhrystone.

Benchmark F enables the auto-inlining of user functions, and G combines the Dhrystone source files into one module. Note that the F and G optimizations are not permitted within the constraints of the Dhrystone benchmark.

The Dhrystone performance is calculated from:

- Dhrystone Iterations per second = $\frac{\text{Processor Clock x Number of Runs}}{\text{Execution Time}}$
- For the result to be valid, the Dhrystone code must be executed for at least two seconds.

The default benchmark D spends approximately 21% of the benchmark cycles executing memcpy, strcpy and strcmp instructions. For benchmark A, this rises to 65%, and for G it is 30%.

Competitive Comparison

"It's very easy to say Dhrystone is not the best benchmark in the world but everyone knows the same tricks and does the same things and that makes it fair again."
Ian Walsh, technical director for MIPS Technologies Europe [4]

Whilst many vendors do understand which 'tricks' can be used to enhance Dhrystone performance, it is not the case that all vendors apply these techniques when measuring Dhrystone. ARM quotes Dhrystone 2.1 figures with strict adherence to the letter and spirit of the rules.

If Dhrystone data must be used for comparison purposes, it is important that the conditions of the benchmark are well understood, including:

1. Which Dhrystone version was used?
2. Which Dhrystone source code was used (ANSI, unmodified K&R)?
3. How many compilation modules were used (one merged or two separate modules)?
4. Which inlining settings are applied?
5. Which C libraries are used?
6. Which tool chain (compiler version, options, linker) was used?
7. What is the CPU's clock speed?

8. Which platform was used (simulator, real hardware)?
9. Which reference number (if other than VAX 11-780 = 1757) was used?

ARM's Dhrystone measurements are made under the following conditions:

- Dhrystone version 2.1
- Source code: Unmodified K&R
- Compilation modules: two
- Inlining settings: -Onoinline option set.
- C libraries: unmodified C libraries supplied with ADS.
- Tool chain: ADS 1.2
- Platform: ARMulator cycle-accurate instruction set simulator that is part of ADS.

Summary

When first released, the Dhrystone benchmark fulfilled a useful function – at least it gave an alternative indicator to vendors' literal MIPS ratings. However, almost twenty years later, there are undoubtedly better benchmarks available for measuring processor performance.

Code profiling demonstrates that Dhrystone is not a particularly good predictor of real-world performance, especially for complex embedded systems. The benchmark's susceptibility to being optimized, yielding large variations in results, is a considerable weakness if the intent is to use the numbers for comparative purposes. ARM's own measurements, presented here, demonstrate that each method of optimization can increase the apparent DMIPS/MHz performance, and when combined can improve the Dhrystone score by up to three times.

ARM's customers still occasionally ask for Dhrystone MIPS data, and so ARM will continue to provide these numbers. However, ARM does not recommend that Dhrystone results be used as part of any embedded processor evaluation exercise, due to its many noted deficiencies. Where ARM provides a Dhrystone figure, it will be based on its standard tool suite under the conditions outlined above, and will therefore be 100% publicly and independently reproducible.

In general, more representative data will be obtained by using a suite of benchmark data. First, this enables benchmarks to reflect the requirements of specific application domains. Second, the effects of any one optimization will be mitigated by the benchmark having a wide spread of algorithms and coding styles. Results based on such an approach are more likely to be representative of processor performance than compiler or tool chain performance.

ARM fully supports and actively participates in the EEMBC process [3]. EEMBC consists of a set of five application suites targeting automotive/industrial, consumer, networking, office automation and telecommunications. The results of any benchmark must be officially certified before they can be published, ensuring a high degree of consistency and reliability in the results. In future, more diverse and representative benchmarks such as EEMBC will better help customers in their product selection processes.

ARM is driving its processor roadmap based on the real needs of applications, by balancing performance needs with the requirement to deliver low power and gate counts. Recent architectural enhancements in ARMv6 include improved memory management, acceleration of DSP and multimedia operations, and in direct Java bytecode execution. To provide useful and meaningful information, in the future, standard benchmarking suites will have to keep up with innovations in processor design.

References

[1] “Understanding Variations in Dhrystone Performance”
By Reinhold P. Weicker, Siemens AG, AUT E 51, Erlangen
April 1989 Microprocessor Report, May 1989 (Editor: M. Slater), pp. 16-17

[2] Reinhold P. Weicker: Dhrystone: A Synthetic Systems Programming Benchmark. Communications of the ACM 27, 10 (Oct. 1984), 1013-1030

[3] EEMBC benchmarking consortium
www.eembc.org

[4] “Figure fight”
By Nick Flaherty
16th July 2001 EETimes, UK
http://www.electronicstimes.com/story/OEG20010711S0003