ARM Compiler (RVDS)

Meet The New ARM Compiler 5

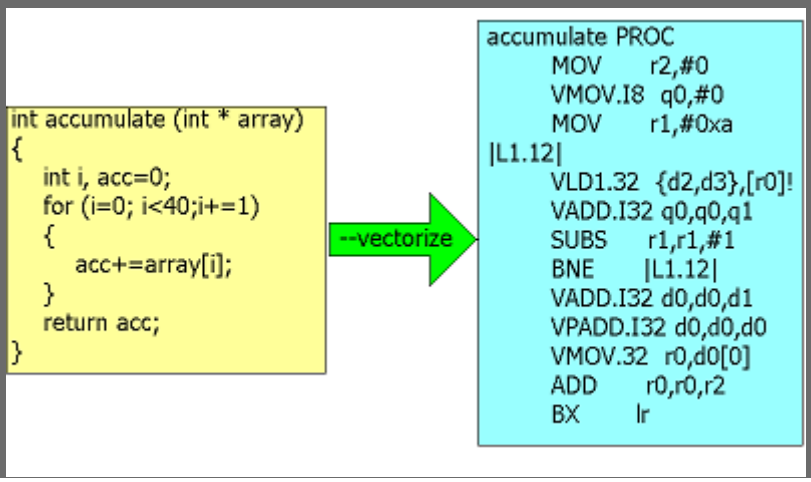
The latest version of the code generation toolchain from ARM is available now in the ARM DS-5™ Professional Edition. The ARM Compiler 5supersedes RVDS 4.1, and offers support accurate, secure and optimized code for the widest range of ARM processors.

The ARM® Compiler in ARM RVDS™ 4.1 is the only commercial compiler co-developed with the ARM processors and specifically designed to optimally support the ARM architecture. It is the result of 20 years of development and is recognized as the industry standard C and C++ compiler for building applications targeting the ARM, Thumb, Thumb-2, VFP and NEON instruction sets.

- ARM processors are designed to best execute code generated by the ARM Compiler
- ARM Compiler enables the new processor features in all the ARM processors
- Profile-Driven Compilation automatically optimizes code based on runtime information
- The ARM Compiler supports building of Symbian OS and ARM Linux applications and libraries

The RVDS 4.1 compiler reduces the best codesize by up to 5% and improves the best performance by 10-15% when compared to RVDS 4.0.

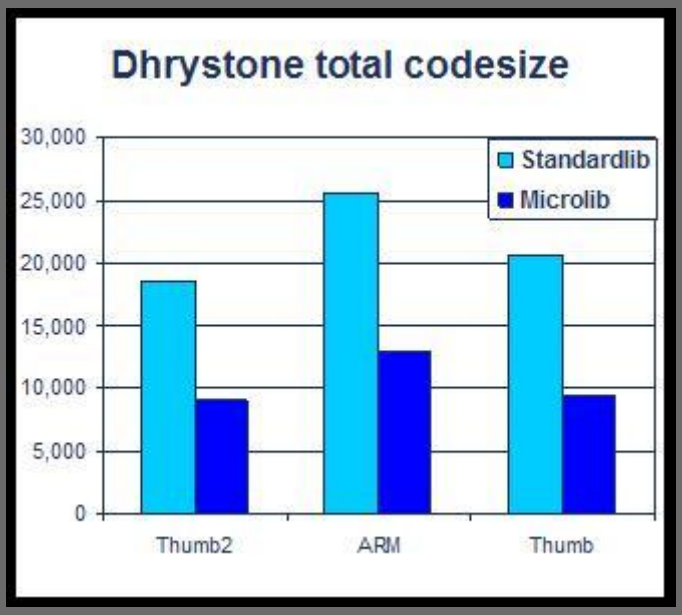
In addition to the ARM Compiler, the state-of-the-art NEON Vectorizing Compiler, which is part of RVDS Professional, enables the automatic generation of ARM NEON SIMD code sequences from standard C and C++ code. The NEON Vectorizing Compiler can speed-up critical multi-media kernels by 4X, resulting in a 2X overall application performance increase.



Development for Severely Cost-Sensitive Devices

To further improve code density for the range of ARM processor-based applications, the ARM Compiler features an optional microlib C library (a subset of the ISO standard C runtime library), which has been minimized in size for microcontroller applications. The microlib C library achieves a 92 percent reduction in runtime library code size.

When combined with a Cortex-M class processor, the microlib C library provides a completely C-based development environment without the need to revert to assembly language - even for interrupt service routines. This removes the need for specific knowledge of the ARM architecture.



Memory savings provided by the microlib C library will depend on the application, from ‘hello world’ (97 percent reduction) to more complex applications such as Dhrystone (50 percent reduction).

Reducing Risk for Linux Application Development

The ABI for the ARM architecture created by the ARM Compiler team is also implemented in the GNU compiler for ARM. This enables the creation and use of libraries that can be shared between development environments.

The compatibility between the GNU Compiler and the ARM Compiler means that Linux applications can be built using the ARM Compiler. The ARM Compiler provides your development team with high-quality support and the performance, stability and code size benefits you expect from a quality commercial toolkit.

Accurate Code Generation

Many algorithms are now automatically generated from mathematical modeling software using floating point representations for data. Therefore, the accuracy of IEEE bit representation and floating point representations is essential for correct operation of the generated code. The default library selected by the ARM Compiler offers a full complement of C/C++ functionality, including C++ exception handling and IEEE 754 floating point support. The compiler can optionally generate code to use either software floating point or any of the ARM hardware floating point units. Independent of the method used, the compiler generates full IEEE 754-compliant code. This means that your application will generate exactly the same data regardless of the target processor, speeding porting from one device to another.

Targeting Advanced Maths and DSP-Style Solutions

Intrinsic functions provide support for common code sequences or instructions that do not map well onto high-level languages.

- ETSI intrinsics provide telecom primitives, which are used in a number of example algorithms
- TI C55 intrinsics provide for support for algorithms written to exploit TI-specific extensions
- Cortex-M4 intrinsics for targeting the onboard DSP

- Other intrinsics allow access to all ARM hardware instructions not easily accessible from C, reducing the need to write code in assembly language

Where possible, intrinsics are emulated on early processors.