Enabling Java for Windows on Arm64

Porting Java to the AArch64 architecture is a JEP 388 success story.

by Monica Beckwith

March 12, 2021

In June 2020, my team at Microsoft announced its intention to provide a working port of OpenJDK for Windows on the Arm64 platform.

The OpenJDK community welcomed this plan with open arms and helped guide my project team to draft a Java Enhancement Proposal (JEP) for the project. As soon as JEP 388 was drafted, my team also drafted an umbrella bug report to list the working patch’s implementation details.

All change requests were then provided to the OpenJDK community via the existing AArch64-port repository on GitHub, which helped the team gather reviews and approvals.

Very soon, the port became a candidate, and then it was proposed to target JDK 16. At the same time, the team started pull request 212 on GitHub and worked with the community to integrate the port into the mainline. Now this port is a part of JDK 16!

In this article, I will provide a few details with respect to OpenJDK, Arm64, and my team’s port to give a deeper description of the community and the development effort.

Introduction to Arm64

Arm is a RISC (reduced instruction set computer) architecture in which the instruction set is highly optimized and the architecture provides a large number of registers. With Arm64, also known as AArch64, Arm introduced 64-bit int registers, data, and pointers. This is called the Armv8-A instruction set architecture (ISA).
Armv8-A ISA is still a weak memory model, but it provides multiple-copy atomicity. A weaker memory model implies that barriers (that is, fences) are needed to guarantee access ordering.

**Porting the JDK**

OpenJDK has a group called the Porters Group that helps extend OpenJDK on different processor architectures and operating systems. Whenever a new platform is available to developers and end users, someone from the OpenJDK community will step up to bring OpenJDK to the new platform.

Porting helps bring a Java Runtime Environment (JRE) to end users to run their Java applications on the new platform. Also, porters help developers develop Java applications by enabling JDK development on the new platform. A JDK is a superset of a JRE because it includes tools and utilities for development and debugging (see Figure 1).

![Figure 1. A JRE is composed of many parts.](image)

**OpenJDK on Windows on Arm64**

Microsoft has a big focus on enabling Windows on Arm (WoA), which is evident from Microsoft's developer-centric offerings such as the Surface Pro X computer.

Although Arm64 has a vibrant ecosystem, there was no Java offering for Surface Pro X devices. Given the Windows developer base, Microsoft thought this was an excellent
opportunity for it to make a significant contribution to the OpenJDK ecosystem.

Most of the OpenJDK HotSpot source code is non-OS and non-architecture specific. All architecture-specific code resides under the `cpu` directory in directories named for the architecture type, such as AArch64, x86, and PPC. Similarly, OS-specific code resides in the `os` directory, and OS and architecture code resides in the `os_cpu` directory.

Besides adding platform-specific additions (in the `windows_aarch64` directory under the `os_cpu` tree), the team also modified code in the `linux_aarch64` directory as well as the `windows_x86-64` directory to make the original code universal in its applicability.

In addition, the team extended its functional, performance, and regression tests to Linux+AArch64 and Windows+x86-64 systems to cover all the changes. This helped assure the community that these patches do not perturb or pose any regression in the existing codebase. The team ran comparative performance and regression tests when it added incremental patch sets documented in the `openjdk-aarch64` GitHub repository.

**Conclusion**

You can see the results of this project by visiting the JEP 388 Windows/AArch64 Port web page.

**Dig deeper**

- Java on Arm processors: Understanding AArch64 vs. x86
- Update on 64-bit ARM support for Oracle OpenJDK and Oracle JDK
- Learn the architecture: AArch64 instruction set architecture

---

**Monica Beckwith**

Monica Beckwith, a principal software engineer at Microsoft, is a Java Champion and a JavaOne Rock Star. She has made many performance contributions to the Java HotSpot VM by identifying the need for a NUMA-aware allocator and allocation patterns, reduction of redundant instructions, reduction of the Java object header, prefetching patterns, redundant array checks in a loop, and various other optimizations for the just-in-time compiler, the generated code, the JVM heuristics, and garbage collection and garbage collectors. Prior to joining Microsoft,
Beckwith was the JVM performance architect at Arm. Her past also includes leading Oracle's G1 garbage collector performance team.