RISC-V and UEFI[1]

Dong Wei, Vincent Zimmer[2]
Hewlett-Packard Intel Corporation
May 27, 2015

Abstract

RISC-V describes a compos-able architecture for various instruction set and system profiles to have a central processing unit (CPU) in a system on a chip (SOC). A CPU in isolation cannot provide a full provide solution to support running various operating systems. To that end, the CPU needs to be situated inside of a platform. And for this SOC-platform to launch operating systems (OS), some hand-off between the platform and OS loader is required. To avoid having to hard-code platform details into the OS, industry standards for platform booting under the umbrella of the Unified Extensible Firmware Interface (UEFI) Forum, such as the main UEFI Specification and ACPI interface come into play. And to build out the UEFI interfaces, open source code base technologies such as EDK II provide potential building blocks. This paper will discuss how EDK II can be used to provide a binding to RISC-V, including the approach and challenges therein.

1 RISC-V Background

Details on the RISC-V instruction set and system features can be found at [1]. RISC-V builds on the heritage of the classic RISC CPU's and features instruction set architecture widths, including 32-bit, 64-bit, and 128-bit. The design is open such that the elements can be built into SOC's that scale from small Internet of Things (IOT) to scale-out servers that need huge address space access.

2 UEFI

EDKII [3] is an open-source implementation of the UEFI specification. The UEFI Forum includes both the main UEFI Specification that is intended to provide an abstraction between the operating system and the underlying firmware. The UEFI Platform Initialization (PI) specifications, in turn, describe a modular means by which to implement the UEFI Specification.

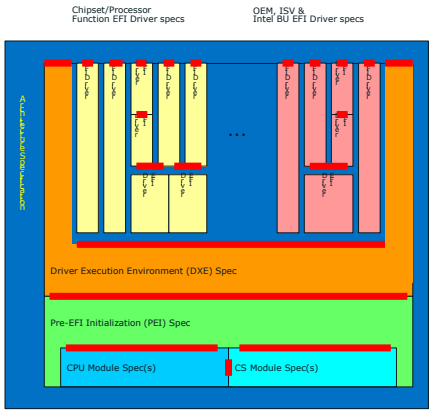

Figure 1 System Diagram of UEFI and PI APIs

Figure 1 above shows the layering with the UEFI API's as red lines at the top and the underlying PI API's at the bottom.

The open source implementation is generic, and there are various CPU-specific elements, such as https://github.com/tianocore/edk2/tree/master/ArmPkg for ARM Ltd. CPU's, both 32-bit and Aarch64, and https://github.com/tianocore/edk2/tree/master/UefiCpuPkg for Intel-architecture based 32-bit and 64-bit CPU's. And overview of silicon support can be found in [5].

Chapter 2 of the UEFI 2.5 specification describes the specifics of a CPU binding, including the machine state for a given CPU. The UEFI PI specification describes the various architectural protocols (Aps) that must be provided in order to make the generic DXE main work with different platforms and CPU architectures. In general, these AP's provide support for a timer, CPU interrupt, and non-volatile storage capability. UEFI in general has a polled driver model so only a timer tick is necessary from the platform.

The EDK II open source implementation is written in

[1] This paper has been submitted to the 2nd RISC-V workshop
[2] The authors can be reached at vincent.zimmer@intel.com and dong.w

pure ANSI C and works with GCC, Intel Compiler, Microsoft Compiler, Clang LLVM/XCode, and many other tool chains.

3 RISC-V and UEFI

To support RISC-V, the approach would be to create a RiscVPkg and RiscVPlatformPkg. The latter could be based upon a software simulator like QEMU.

The value of having UEFI support is that UEFI and ACPI, with the hardware reduced bit of the latter, will allow for re-use of upstream operating support, such as the Linux kernel. Instead of having sub-architectures for each SOC, there can be a single kernel that parameterizes the platform differences with UEFI and ACPI. This is how the ARM community went from many sub-archs to a single CPU support directory, akin to the arch/x86 for the Intel-architecture based systems.

4 Related Work

Although there is not a RISC-V binding in chapter 2 of the UEFI 2.5 specification, other CPU's, such as MIPS [4], have been ported to UEFI.

5 Conclusion

Ours is a work in progress. We have discussed industry-standard firmware and how to prepare for a new architecture, such as RISC-V.

Acknowledgments

We would like to thank the UEFI Forum members for the work on the UEFI, PI, and ACPI specification, along with the community for work on the EDK II implementation.

6 References

[1] RISC-V website http://www.risc-v.org/.

[2] UEFI website http://www.uefi.org/.

[3] EDK II project http://www.tianocore.org/.

[4] MIPS EDK port http://sourceforge.net/projects/efi-mips/

[5] Isaac Oram, Tim Lewis (Phoenix), Vincent Zimmer, "Silicon Enabling in a Modular Architecture," in *Intel Technology Journal - UEFI Today: Boostrapping the Continuum*, Volume 15, Issue 1, pp. 22-39, October 2011, ISBN 978-1-934053-43-0, ISSN 1535-864X http://www.intel.com/content/www/us/en/research/intel-technology-journal/2011-volume-15-issue-01-intel-technology-journal.html