Science of Computer Programming 187 (2020) 102371

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Formalizing SPARCv8 instruction set architecture in Coq ☆

Jiawei Wang [a], Ming Fu [b], Lei Qiao [c], Xinyu Feng [d,*]

[a] University of Science and Technology of China, Hefei, China
[b] Huawei Technologies Co., Ltd., Shanghai, China
[c] Beijing Institute of Control Engineering, Beijing, China
[d] Nanjing University, Nanjing, China

ARTICLE INFO

Article history:
Received 31 March 2018
Received in revised form 15 April 2019
Accepted 12 November 2019
Available online 19 November 2019

Keywords:
SPARCv8
Coq
Verification
Operational semantics

ABSTRACT

The SPARCv8 instruction set architecture (ISA) has been widely used in various processors for workstations, embedded systems, and space missions. In order to formally verify the correctness of embedded operating systems running on SPARCv8 processors, one has to formalize the semantics of SPARCv8 ISA. In this article, we present our formalization of SPARCv8 ISA, which is faithful to the realistic design of SPARCv8. We also prove the determinacy and isolation properties with respect to the operational semantics of our formal model. In addition, we have verified that two trap handlers handling window overflow and window underflow satisfy the user's specifications based on our formal model. All of the formalization and proofs have been mechanized in Coq.

1. Introduction

Computer systems have been widely used in national defense, finance and other fields. Building high-confidence systems plays a significant role in the development of computer systems. Operating system kernel is the most foundational software of computer systems, and its reliability is the key in building high-confidence computer system.

In aerospace and other security areas, the underlying operating system is usually implemented in C and assembly languages. In existing OS verification projects, *e.g.*, CertiμC/OS-II [1] and seL4 [2], the assembly code is usually not modeled in order to simplify the formalization of the target machine. They use abstract specifications to describe the behavior of the assembly code to avoid exposing the details of underlying machines, *e.g.*, registers and stacks. Therefore, the assembly code in those kernels is not actually verified. To verify whether the assembly code satisfies its specifications, it is inevitable to formalize the semantics of the assembly instructions.

As a highly efficient and reliable microprocessor, the SPARCv8 [3] instruction set architecture has been widely used in various processors for workstations, embedded systems, and space missions. For instance, SpaceOS [4] running on SPARCv8 processors is an embedded operating system developed by Beijing Institute of Control Engineering (BICE) and deployed in the central computer of Chang'e-3 lunar exploration mission. On the one hand, to formally verify SpaceOS, we need to formalize the SPARCv8 instruction set and build the mathematical semantic model of the assembly instructions. On the other hand, to ensure the consistency between the behavior of the target assembly code and the C source code, we hope to use the certified compiler CompCert [5] to compile SpaceOS. However, CompCert does not support SPARCv8 at the backend.

☆ This work is supported in part by grants from National Natural Science Foundation of China (NSFC) under Grant No. 61632005.
* Corresponding author.
E-mail address: xyfeng@nju.edu.cn (X. Feng).

https://doi.org/10.1016/j.scico.2019.102371
0167-6423/

Extending CompCert to support SPARCv8 also requires us to formalize the SPARCv8 instruction set architecture. In this paper, we make the following contributions:

- We formalize the operational semantics of the SPARCv8 ISA. Our formal model is faithful to the behaviors of the instructions described in the SPARCv8 manual [3], including all the integer units with most of the features in SPARCv8, *e.g.*, windowed registers, delayed control transfer, and interrupts and traps.
- We prove that the operational semantics satisfies the determinacy property, and the execution in the user mode or supervisor mode satisfies the isolation property.
- We take the trap handlers for window overflow and window underflow as examples, and give their pre-condition and post-condition to specify the expected behaviors. Like proving programs with Hoare triples [6], we prove that these trap handlers satisfy the given pre-/post-conditions and do not throw any exceptions.
- All of the formalization and proofs have been mechanized in Coq [7]. They contain around 14000 lines of Coq scripts in total (measured by cloc [8]). The source code can be accessed via the link [9].

This article extends the conference paper in SETTA 2017 [10]. First, we give more details about our model, including the definitions of the windows rotation operation, delayed writes, trap and abort handling, *etc.*. Second, we present operational semantics rules for more SPARCv8 instructions, including some delayed transfer rules, abort rules, *etc.*. Third, we provide more detailed explanation about why the window overflow occurs, and how the window overflow trap handler handles it. Finally, to ease the proof burden, we give a tool called 'coq2smt' [11] that can translate lemmas about arithmetic propositions from Coq into SMT solvers such as Z3 [12], and use it to solve these lemmas. The tool contains around 6000 lines of Coq and Ocaml code in total (measured by cloc [8]). We use this tool to verify the window underflow trap handler, and it contains 5500 lines of Coq scripts.

Related work Fox and Myreen gave the ARMv7 ISA model [13]. They used monadic specification and formalized the instruction decoding and operational semantics. Narges Khakpour et al. proved some security properties of ARMv7 in the proof assistant tool HOL4, including the kernel security property, user mode isolation property, and so on. Andrew Kennedy et al. formalized the subset of x86 in Coq [14], and they used type classes, notations and the mathematics library Ssreflect [15]. The CompCert compiler also has the formal modeling of ARM and x86. There are lots of modeling work related to the x86 and ARM, but due to the specific features of SPARCv8, these x86 and ARM ISA models can not be used directly for the SPARCv8 ISA.

Zhe Hou et al. modeled the SPARCv8 ISA in the proof assistant Isabelle [16], which is close to our work. But their work is focused on the SPARCv8 processor itself, instead of the assembly code running on it. To verify the assembly code, we need a better definition on the syntax and the operational semantics. And the definition of machine state needs to be hierarchical and easy to use when we verify the code running on it. Additionally, they did not model the interrupt feature in SPARCv8, hence their model could not describe the non-determinism of the operational semantics caused by interrupts. Besides, our formalization of the SPARCv8 ISA is implemented in Coq, while CompCert is implemented in Coq too. We can use our Coq implementation to extend the CompCert at the backend to support SPARCv8 in the future.

There are several tools that integrate SMT solvers into Coq [17–19], but none of them are suitable for proving low-level code. To prove the low-level code, it is inevitable to deal with different bits of integers and various arithmetical operations. Base on [19], we develop a tool called 'coq2smt' that can translate dozens of lemmas for arithmetical operations on 8, 16, 32 and 64 bits integers in Coq to the SMT solver and solve it.

There are some other verification work at assembly level [20–22], which give the formal models of different subsets of x86 instructions and the behavior of the x86 interrupt management. They mainly study the verification techniques of assembly code, while the instruction set is relatively small. We formalized the SPARCv8 ISA by considering all the features of the integer unit of SPARCv8 [3]. In the next section, we give a brief overview of these features.

2. Overview of SPARCv8 ISA

The Scalable Processor Architecture (SPARC) is a reduced instruction set computing (RISC) instruction set architecture (ISA) originally developed by Sun Microsystems. It is widely used in the electronic systems of space devices for its high performance, high reliability and low power consumption. For example, LEON3 [23], a SPARCv8 architecture-based processor, developed by the European Space Research and Technology Center, is widely used in application-specific integrated circuits.

Compared to other architectures, SPARCv8 has the following unique mechanisms:

- A variety of control-transfer instructions (CTIs) and annulled delay instructions for more flexible function jumps.
- The register window and window rotation mechanism for swapping context more efficiently.
- Two modes, user mode and supervisor mode, for separating the application code and operating system code at the physical level.
- A variety of traps for swapping modes through a special trap table that contains the first 4 instructions of each trap handler.
- Delayed-write mechanism for delaying the execution of register write operation for several cycles.