**稿件编号：JCST-2004-10536.R1**

**中文题目：SPARCv8代码的模块化验证**

**作者中文姓名：查君鹏，冯新宇，乔磊**

**中文关键词：SPARCv8，汇编代码验证，上下文切换，Coq，精化验证**

**中文长摘要:**

1、研究背景（context）

系统软件（如：操作系统、虚拟机等）常使用内嵌汇编代码实现与底层硬件平台之间的交互，所以汇编代码的正确性对于保证整个系统的正确性是非常重要的。而形式化验证技术基于严格的数学理论和方法，能够为程序的正确性提供强有力的保证。在某航天嵌入式操作系统的形式化验证项目中，该操作系统的内嵌汇编代码使用SPARCv8 指令集编写，但项目前期工作中所使用的操作系统验证框架没有提供验证汇编代码的方法，而是定义了一些抽象的原子操作（称为“抽象汇编原语”）来替代具体的汇编程序。SPARC,即可扩充处理器架（Scalable Processor ARChitecture），是国际上流行的RISC处理器体系结构之一。SPARCv7/v8是目前嵌入式控制系统常用的处理器标准版本，并在各种处理器、工作站以及嵌入式系统中得到广泛的应用。

2、目的（Objective）和方法（Method）

为了完善当前某航天嵌入式操作系统验证工作中内嵌SPARCv8汇编程序没有验证的问题，我们希望为SPARCv8汇编程序设计一套验证逻辑，该程序验证逻辑能够保证通过验证的SPARCv8汇编程序与其对应的抽象汇编原语之间具有上下文精化关系（contextual refinement），即：在任意上下文下，调用抽象汇编原语SPARCv8实现的程序都能保持调用抽象原语的程序的程序行为。公式(1)展示了上下文精化关系，这里和分别表示抽象汇编原语及其对应的SPARCv8实现，而表示调用抽象汇编原语及其对应的SPARCv8程序的上下文。

(1)

我们所验证的航天嵌入式操作系统由C语言和SPARCv8指令集实现。但若使用C程序作为上述上下文，需要将C语言和SPARCv8汇编语言在统一的机器模型上建模。本文工作侧重于SPARCv8汇编程序的形式化验证，所以我们使用图1展示的方法来避免如何将C语言和SPARCv8汇编语言在统一机器模型上建模的问题。

图1 证明方法

图1中，我们使用表示调用抽象原语的操作系统程序，其中表示C语言程序。使用C程序语义执行（表示为）。编译器（）将C语言程序编译为SPARCv8程序。由步骤所示，我们假设编译器的正确性能够保证程序保持源程序的程序行为。我们将称为Pseudo-SPARCv8程序（可以调用抽象汇编原语的SPARCv8程序，语义表示为）。我们用表示定义在Pseudo-SPARCv8程序语义下的抽象汇编原语。步骤，我们定义SPARCv8程序验证逻辑（表示为）去形式化验证抽象汇编原语SPARCv8实现的正确性，验证逻辑保证了调用抽象汇编原语SPARCv8实现的SPARCv8程序保持调用抽象汇编原语的Pseudo-SPARCv8程序（表示为），即：。

本工作专注于步骤，而将步骤作为以后的工作。

为SPARCv8程序设计验证逻辑并不是一件容易的事情，主要是因为SPARCv8程序具有一些特性，这些特性可以总结为：

1. 延时跳转：SPARCv8程序在执行jmp（无条件跳转）和retl（函数返回）等跳转指令时，不会立即发生跳转，而会延时一个执行周期。
2. 特殊寄存器的延时写入：SPARCv8中存在一类特殊寄存器，对特殊寄存器的修改不会立即生效，而是会延时个指令周期之后再生效。是一个预定义的0到3之间的系统参数。
3. 寄存器窗口机制：SPARCv8 使用寄存器窗口和窗口旋转机制来实现程序上下文管理，这样可以避免将当前方法的运行时上下文直接保存到内存的栈空间，从而提高程序的执行效率。

为了支持SPARCv8三个主要特性的验证，我们在定义SPARCv8程序验证逻辑时分别做了如下工作：

1. 延时跳转：我们重新定义了基本代码块，将jmp和retl等延时跳转指令及其直接后继指令一起视为一个基本代码块的结尾来解决 SPARCv8 中的延时跳转问题。
2. 特殊寄存器的延时写入：我们定义了一种新的断言来描述特殊寄存器的状态，表示特殊寄存器的值将会在至多个周期之后为。
3. 寄存器窗口机制：SPARCv8主要通过save和restore指令来操作寄存器窗口来实现高效的函数执行上下文管理。我们为save和restore指令设计了相应的推理规则。

3、结果（Result & Findings）

首先，我们为SPARCv8汇编程序设计了一套精化验证逻辑，该验证逻辑：

1. 支持SPARCv8三个主要特性（延时跳转、特殊寄存器的延时写入以及寄存器窗口机制）的验证。
2. 支持函数调用的模块化验证。
3. 能够保证通过验证的抽象汇编原语的SPARCv8实现与该抽象汇编原语之间具有上下文精化关系。
4. 我们赋予了验证规则直接风格的（direct-style）基于语义的解释。方便我们在上下文精化关系的证明中，定义抽象汇编原语SPARCv8实现与该抽象汇编原语之间的模拟关系。

其次，我们定义了Pseudo-SPARCv8语言，用于实现上下文精化关系验证工作中的高层规范（high-level specification）。

最后，我们使用我们定义的SPARCv8程序验证逻辑，验证了一个SPARCv8实现的任务上下文切换程序与抽象switch原语之间具有上下文精化关系。

4、结论（Conclusions）

我们为SPARCv8汇编程序设计了一套精化验证逻辑，该验证逻辑支持 SPARCv8 汇编程序中函数调用的模块化验证以及SPARCv8 的三个主要特性：延时跳转、特殊寄存器的延时写入，以及寄存器窗口机制的验证，并能够保证通过验证的抽象汇编原语SPARCv8实现与该抽象汇编原语之间具有上下文精化关系。我们用该验证逻辑验证了一个SPARCv8实现的任务上下文切换程序与抽象switch原语之间具有上下文精化关系。

不过，当前工作仍有需要在未来工作中去完善的地方。首先，当前工作所定义的SPARCv8机器模型没有考虑中断；其次，图 1所展示的步骤还没有完成，即：编译器保证编译得到的Pseudo-SPARCv8程序能够保持调用抽象汇编原语的C语言源程序的程序行为；最后，应用一些自动证明策略（如：Coq Tactic）和证明工具（如：Z3 SMT solver）来提高证明的效率。