Desenvolvimento de Testes de Qualidade para Verificação de Processadores RISC-V

Palavras-Chave: RISC-V, GERAÇÃO DE TESTES, VERIFICAÇÃO DE HARDWARE

Autores(as):
Enrico Fernandes, IC – UNICAMP
Prof. Dr. Rodolfo Azevedo, IC – UNICAMP

INTRODUÇÃO:

A diversidade de implementações de processadores baseados na arquitetura RISC-V presentes tem criado desafios significativos para a verificação de conformidade com as especificações.

Esta diversidade de implementações, embora boa para inovação e competitividade no mercado, apresenta desafios únicos para metodologias de verificação tradicionais. Estudos recentes indicam que mais de 200 implementações diferentes para processadores RISC-V estão atualmente disponíveis, abordando desde microcontroladores embarcados até processadores de alto desempenho para computação em nuvem [1,2].

O problema torna-se particularmente complexo quando consideramos que cada implementação pode ter suporte variável para extensões opcionais da ISA. Metodologias tradicionais de verificação baseadas em conjuntos de testes estáticos se mostram inadequadas para testar adequadamente todas essas implementações simultaneamente.

Este trabalho apresenta o desenvolvimento de um framework completo para geração automatizada e validação de casos de teste para verificação de processadores RISC-V, abordando sistematicamente as limitações identificadas na literatura através de uma arquitetura modular extensível.

METODOLOGIA:

O desenvolvimento do framework seguiu uma abordagem modular, implementada em Python. A metodologia adotada baseou-se em iterações incrementais para validação contínua dos componentes.

A fase inicial envolveu um estudo aprofundado na especificação RISC-V [3], do conceito de Profiles em RISC-V e quais profiles existem atualmente[4], além de revisão bibliográfica de metodologias existentes para verificação de processadores [5,6]. Foram identificados quatro requisitos

essenciais: (1) geração de casos de teste que explorem instruções diversas; (2) validação sintática e semântica dos testes gerados; (3) classificação de casos de teste por extensões; (4) determinação de compatibilidade entre conjuntos de instruções e perfis específicos de processadores.

O programa foi estruturado em diversos módulos, com funções bem definidas: o módulo de coleta de dados adquire os dados mais recentes de repositórios oficias RISC-V para construir um banco de dados com as especificações para as instruções ratificadas e com os profiles disponiveis, esses dados serão utilizadas para saber quais instruções existem, a que grupo pertencem e quais extensões são permitidas por cada profile.

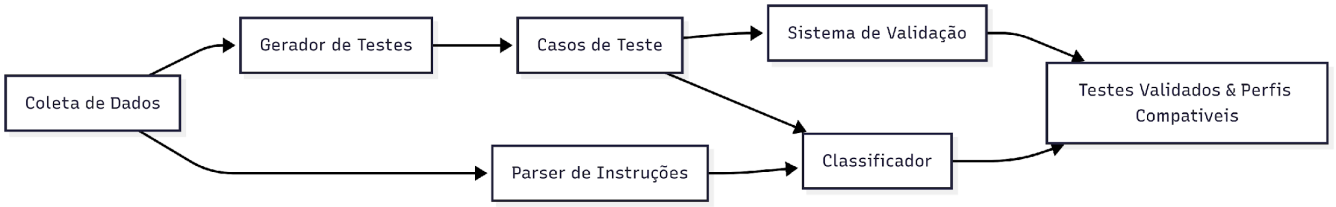

Grafico 1 – Gráfico de fluxo de informação entre módulos desenvolvidos

O módulo de geração de testes implementa algoritmos de geração pseudo-aleatória de testes com restrições baseadas em regras gramaticais específicas para cada formato de instrução possível para RISC-V (instruções R, I, S, B, U e J).

O módulo de parsing de instruções decodifica instruções e permite classificação de instruções presentes em código assembly em binário/hexadecimal para processadores RISC-V. O módulo de classificação de profiles permite ao usuário analisar um teste ou um conjunto de testes e determinar quais profiles RISC-V são compatíveis com o grupo de instruções presentes naquele grupo de testes com uma precisão superior a 85%. Há também um módulo de gerenciamento de profiles que permite a construção de novos profiles para o usuário.

O algoritmo de geração de casos de teste usa restrições para garantir validade sintática e semântica dos casos gerados:

• **Geração de Registradores:** Aplicação de limites específicos (0 ≤ reg ≤ 31, reg ≠ x0 para registradores de destino, etc).

• **Sistema de Templates**: Implementação de contexto para geração de sintaxe assembly válida, com substituição parametrizada baseada em expressões.

Tipo de Instrução	**Instrução**	**Restrições**	**Exemplo Gerado**
R-Type (ADD)	ADD rd, rs1, rs2	rd ≠ x0, rs1/rs2 qualquer	add x5, x12, x7
I-Type (ADDI)	ADDI rd, rs1, imm	imm: -2048 a 2047	addi x3, x8, 1500
B-Type (BEQ)	BEQ rs1, rs2, imm	imm par, múltiplo de 2	beq x1, x2, 16

Tabela 1 – Exemplo de Instruções e Restrições aplicadas para geração

O módulo de validação dos casos de teste gerados possui três camadas de verificação: A validação sintática consiste de um parser para verificação de conformidade com a gramática formal da linguagem assembly RISC-V. A validação semântica consiste na aplicação de regras semânticas específicas da ISA, como verificação de ranges de valores e restrições de uso de registradores especiais. E a validação por compilação consiste na compilação desses testes utilizando a ferramenta oficial GNU RISC-V (riscv32-unknown-elf-as) para verificação de compatibilidade, garantindo que todos os casos de teste gerados sejam sintaticamente corretos e semanticamente válidos.

RESULTADOS E DISCUSSÃO:

O framework desenvolvido demonstra capacidades operacionais em coleta automática de dados, com o módulo de coleta demonstrando capacidade de baixar e processar especificações para mais de 100 extensões RISC-V, com capacidade de parsing, normalização e indexação para armazenamento.

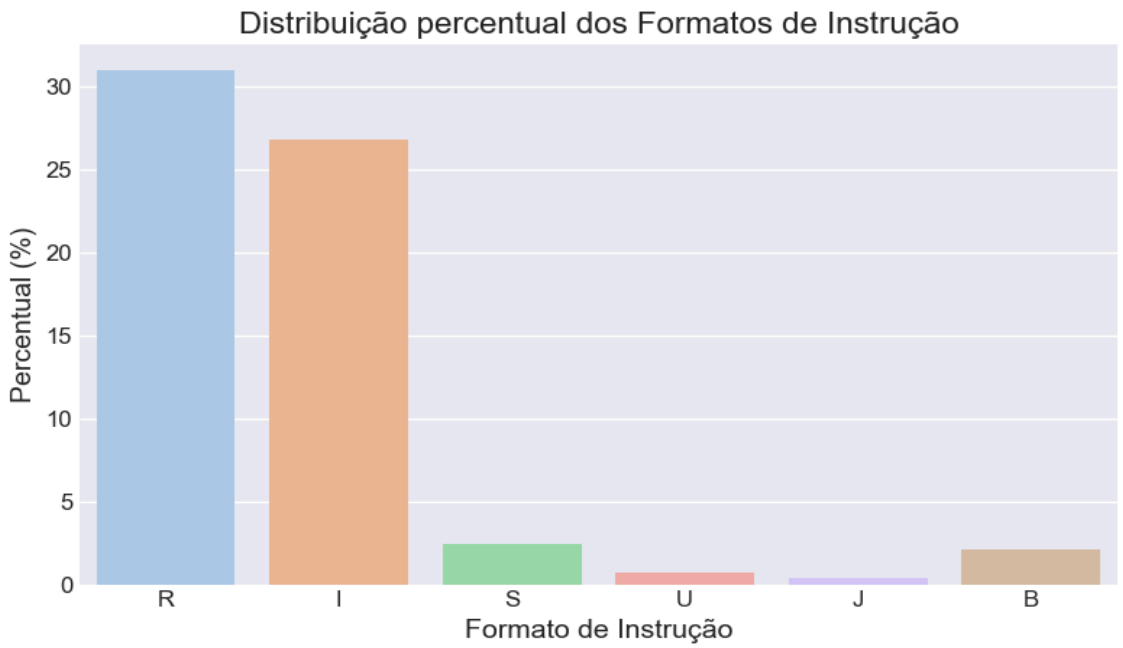

Gráfico 2 – Distribuição percentual de tipos de instrução nos opcodes coletados pelo programa

O algoritmo de geração de casos de teste demonstra complexidade temporal linear com relação ao número de casos de teste solicitados, mantendo taxa de geração constante.

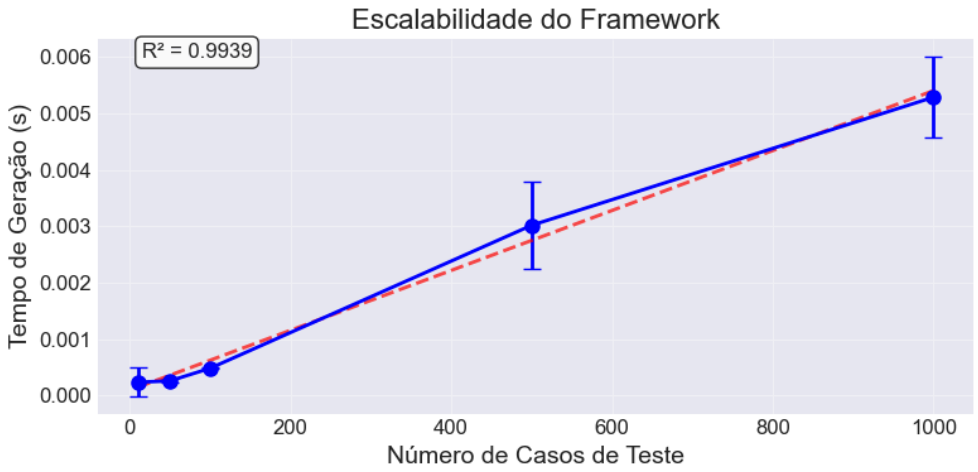

Gráfico 3 – Correlação entre tempo de gerações e n° de casos de teste gerado

A validação experimental desses casos de teste gerados foi conduzida através de cenários representativos de casos de uso reais:

1. Geração de Suítes de Teste em Larga Escala: Foram feitos experimentos com geração de conjuntos de milhares de casos de teste, a geração demonstrou manutenção de performance linear, validando a hipótese de escalabilidade do algoritmo.

2. Análise de Compatibilidade com Profiles: Validação do classificador de instruções e de profiles utilizando conjunto de instruções assembly provenientes de repositórios open-source, resultando em classificação correta de instruções em mais de 85% dos casos, com eficiência comparável a outros métodos disponíveis na literatura [5,6], e retornando quais possíveis profiles seriam capazes de rodar o código verificado.

CONCLUSÕES:

O framework desenvolvido representa uma contribuição significativa para o campo de verificação de processadores RISC-V, demonstrando capacidade de geração de testes extensiva, além de classificação de testes.

A arquitetura modular implementada atende aos requisitos de extensibilidade necessários para acompanhar a evolução contínua da especificação RISC-V, permitindo incorporação de novas extensões, assim como integração em diferentes projetos. A validação cruzada com toolchains oficiais garante conformidade com padrões industriais estabelecidos.

A disponibilização como software livre contribui para o desenvolvimento futuro de tecnologias RISC-V, assim como as métricas de precisão e escalibilidade altas demonstram potencial para aplicação em diversos ambientes de desenvolvimento.

A experiência de pesquisa proporcionou formação sólida em metodologias de engenharia de software, arquitetura RISC-V e verificação formal, estabelecendo fundação adequada para continuidade em programas de pós-graduação na área de sistemas computacionais.

BIBLIOGRAFIA

[1] WATERMAN, Andrew; LEE, Yunsup; PATTERSON, David A.; ASANOVIĆ, Krste. **The RISC-V Instruction Set Manual, Volume I: User-Level ISA**, Document Version 20191213. RISC-V Foundation, 2019.

[2] Enfang Cui, Tianzheng Li, and Qian Wei. **Risc-v instruction set architecture extensions: A survey.**IEEE Access, 11:24696–24711, 2023

[3] RISC-V International. **The RISC-V Instruction Set Manual Volume II: Privileged Architecture**, Document Version 20211203. RISC-V International, 2021.

[4] RISC-V International. **RISC-V Profiles Specification**, Document Version 20220107. RISC-V International, 2022.

[5] HERDT, Vladimir; GROSSE, Daniel; DRECHSLER, Rolf. **Enhanced Virtual Prototyping for RISC-V Processor Verification**. In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2020, pp. 411-416.

[6] ARMSTRONG, Alasdair; BAUEREISS, Thomas; CAMPBELL, Brian; REID, Alastair**. ISA Semantics for ARMv8-A, RISC-V, and CHERI-MIPS**. Proceedings of the ACM on Programming Languages, v. 3, n. POPL, Article 71, 2019.