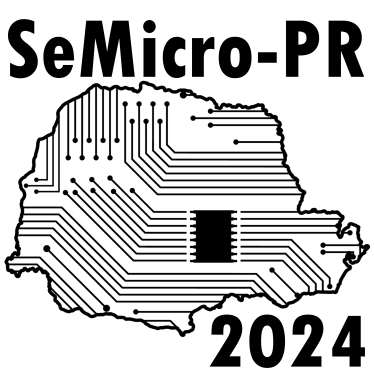

Desenvolvimento do Núcleo de um Simulador Web para a Arquitetura RISC-V

Eduardo M. D. Souza, Thiago H. Rausch,
Cesar A. Zeferino, Douglas R. Melo
LEDS, Universidade do Vale do Itajaí, Itajaí, Brasil
{eduardomichel, thiagorausch}@edu.univali.br, {zeferino, drm}@univali.br

Resumo - A arquitetura RISC-V tem se expandido rapidamente devido ao seu projeto aberto e modular, sendo adotada em diversos setores acadêmicos e industriais. No entanto, ainda enfrenta desafios, especialmente em comparação com arquiteturas estabelecidas como x86 e ARM. Há uma escassez de materiais educacionais, e as ferramentas disponíveis são frequentemente complexas, dificultando o aprendizado da linguagem de montagem RISC-V. Dessa forma, este trabalho apresenta o desenvolvimento de um núcleo de simulador focado no conjunto de instruções RV32I. O núcleo foi desenvolvido em TypeScript e Angular, é composto por Analisador de Código, Montador e Processador, e pode ser integrado a uma interface web, facilitando o aprendizado prático da arquitetura.

I. INTRODUÇÃO

O processador, elemento central dos sistemas computacionais, executa as operações definidas pela arquitetura por meio de um conjunto de instruções conhecido como ISA (Instruction Set Architecture). Dentre as diversas arquiteturas, a classe RISC (Reduced Instruction Set Computer) se destaca por sua simplicidade e eficiência, oferecendo um design que facilita a implementação e o desempenho [1]. Um exemplo dessa classe é o RISC-V, uma arquitetura aberta e gratuita que combina modularidade e flexibilidade, tornando-se uma opção atrativa tanto para sistemas embarcados de baixo consumo de energia quanto para aplicações de computação de alto desempenho [2].

Embora o interesse no RISC-V esteja crescendo, a arquitetura ainda enfrenta desafios para sua adoção mais ampla, especialmente quando comparada a arquiteturas mais consolidadas, como x86 e ARM. Um dos desafios está na disponibilidade de ferramentas educacionais e de suporte para iniciantes [3]. As ferramentas atuais para RISC-V são, muitas vezes, complexas e podem dificultar o aprendizado e a prática da programação em linguagem de montagem, o que limita sua adoção em projetos acadêmicos e industriais [4].

Para enfrentar esses desafios, este trabalho foca no desenvolvimento do núcleo de um simulador para a arquitetura RISC-V, especificamente para o conjunto de instruções RV32I. O núcleo do simulador é composto de três componentes: o Analisador de Código, que interpreta o código-fonte; o Montador, que converte esse código para linguagem de máquina; e o Processador, que executa as instruções geradas. Além desses componentes, o núcleo também inclui o Objeto Final do Programa, que armazena o código binário e as informações do processador necessárias para a execução. O objetivo principal é desenvolver um núcleo funcional e modular para o simulador, que possa ser facilmente integrado a uma interface web no futuro, oferecendo um ambiente interativo para o aprendizado e experimentação com a arquitetura RISC-V. O código será disponibilizado em repositórios colaborativos, permitindo que a comunidade participe de seu desenvolvimento contínuo.

O restante do artigo está organizado da seguinte forma: na Seção II, é apresentado o referencial teórico utilizado. A Seção III descreve os trabalhos relacionados. A Seção IV aborda o desenvolvimento do projeto. Na Seção V, são discutidos os resultados. Por fim, a Seção VI traz a conclusão e sugestões para trabalhos futuros.

II. REFERENCIAL TEÓRICO

O processador é o componente central em computadores, responsável por executar programas e interagir com memória e dispositivos periféricos, impactando o desempenho geral do sistema [5]. O conjunto de instruções (ISA) atua como a interface entre o software e o hardware, definindo as operações que o processador pode executar, como instruções aritméticas, lógicas e de controle de fluxo [6]. Entre as principais arquiteturas de processadores estão as abordagens RISC, com instruções simples, e CISC (Complex Instruction Set Computer), que oferece maior versatilidade por meio de instruções mais complexas [7].

O RISC-V é uma arquitetura de conjunto de instruções originalmente desenvolvida para fins educacionais e de pesquisa, mas que evoluiu para uma arquitetura aberta e gratuita, adequada para implementações industriais [8]. Seu conjunto de instruções base, o RV32I, opera com 32 bits e serve como o núcleo da arquitetura, proporcionando estabilidade para diversas aplicações. A modularidade do RISC-V permite que ele seja adaptado para diferentes finalidades, desde dispositivos embarcados de baixo consumo até sistemas de alto desempenho, sendo suas extensões opcionais o que confere flexibilidade para os desenvolvedores [9].

A simulação é uma técnica que replica o funcionamento de sistemas reais em computadores, fundamental em sistemas onde métodos analíticos não conseguem prever o comportamento com precisão [10]. No caso de processadores, a simulação usa modelos virtuais para avaliar o desempenho e validar diferentes designs arquiteturais, ajudando no desenvolvimento de software e hardware [11]. Simuladores como MARS [12] e SPIM [13] permitem escrever, montar e executar código em linguagem de montagem da arquitetura MIPS, facilitando o estudo e a depuração de algoritmos.

III. TRABALHOS RELACIONADOS

Esta seção apresenta simuladores da arquitetura RISC-V, com ênfase em suas funcionalidades e relação com os objetivos deste trabalho.

O RARS (RISC-V Assembler and Runtime Simulator) é uma ferramenta voltada para o ensino e simulação de programas RISC-V, com suporte para múltiplas extensões da ISA, incluindo RV32I e RV64I. Sua principal limitação é ser restrito a ambientes desktop, sem suporte para versões web ou móveis [14].

Vulcan é um simulador RISC-V voltado para o ensino, com foco na visualização do contador de programa e registradores. No entanto, a interface apresenta limitações de layout, e a ausência de simulação interativa reduz sua aplicabilidade para tarefas mais avançadas [15].

O emulsiV é um simulador visual da arquitetura RISC-V, utilizado principalmente em contextos educacionais. Ele permite visualizar o caminho de dados de um processador, mas sua interface e a documentação limitada podem dificultar a usabilidade para novos usuários [16].

Desenvolvido pela Boston University, o BRISC-V é um simulador web educacional que facilita o ensino da linguagem de montagem RISC-V. Ele não permite desfazer passos da simulação e requer o carregamento de código via arquivo, limitando a flexibilidade do usuário [17].

O WebRISC-V é um simulador web focado na simulação do pipeline RISC-V com suporte a RV64I e RV64M. A separação das visualizações de registradores e memória torna a análise simultânea mais difícil, além de exigir uma curva de aprendizado maior [18].

Ripes é um simulador visual integrado para a arquitetura RISC-V, oferecendo recursos como simulação de cache e compilação de programas C. No entanto, sua complexidade pode ser inadequada para usuários que buscam uma experiência mais simplificada [19].

Alguns simuladores apresentam limitações que impactam tanto a usabilidade quanto o desempenho. O RARS exige a instalação do ambiente Java, tornando seu uso mais complexo. O Vulcan, embora eficiente no uso de memória, suporta apenas registradores numéricos e enfrenta problemas de navegação no editor. O emulsiV carece de uma indicação clara de término de execução, dificultando a fluidez na escrita do código. O BRISC-V, apesar de funcional, possui uma interface que complica a visualização e manipulação de registradores e memória. Já o WebRISC-V apresenta uma interface fragmentada, tornando a análise do código mais difícil devido à organização pouco intuitiva.

Este trabalho propõe um núcleo modular para um simulador web, desenvolvido em TypeScript e Angular, com foco na simplicidade e organização. O núcleo é responsável por funções essenciais, como montagem e execução de instruções RISC-V do conjunto RV32I. Sua arquitetura permite execução direta no navegador, sem necessidade de servidores externos ou softwares adicionais, facilitando a integração com uma interface intuitiva que oferecerá recursos como visualização de registradores, memória e execução passo a passo, tornando o aprendizado mais acessível e interativo.

IV. DESENVOLVIMENTO

O desenvolvimento do núcleo do simulador foi realizado utilizando o editor de código VS Code, devido ao seu suporte a diversas extensões para o desenvolvimento em JavaScript e TypeScript. O núcleo foi implementado em Angular na versão 16, um *framework* baseado em TypeScript, que organiza os componentes de forma modular e permite futuras expansões.

O núcleo do simulador é composto por três componentes principais: Analisador de Código, Montador e Processador. O Analisador de Código transforma o código escrito pelo usuário em um objeto intermediário, que contém todas as informações necessárias para a próxima etapa. Esse objeto intermediário é então processado pelo Montador, que o converte em linguagem de máquina. Durante esse processo, é criado o Objeto Final do Programa, que armazena o código binário, os dados do programa e as informações sobre o estado do processador. O Processador, por sua vez, utiliza o Objeto Final do Programa para executar as instruções e simular o comportamento do processador RISC-V. A Figura 1 representa a modelagem do núcleo do simulador, contendo o Analisador de Código, Montador e Processador.

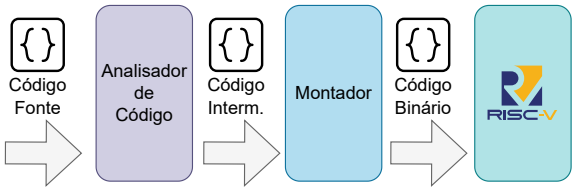

Fig. 1: *Modelagem Geral*

A. Analisador de Código

O Analisador de Código recebe o código escrito pelo usuário e cria um objeto intermediário, que serve como base para o Objeto Final do Programa. Esse objeto intermediário contém as instruções do código-fonte, os dados definidos no segmento de dados e o estado inicial do processador, incluindo memória e registradores. O Analisador processa as instruções e pseudo-instruções, criando a tabela de símbolos e calculando os deslocamentos (*offsets*) necessários. A Figura 2 representa a modelagem do Analisador de Código, mostrando o fluxo de criação do objeto intermediário a partir do Código Fonte.

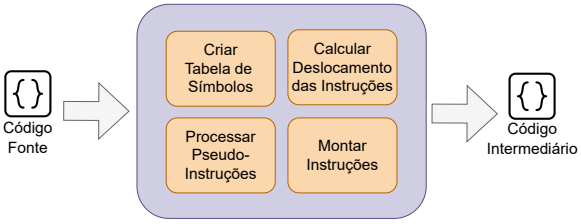

Fig. 2: *Modelagem do Analisador de Código*

B. Montador

Após o Analisador de Código criar o objeto intermediário, o Montador entra em ação para montar o código. Ele percorre as instruções armazenadas no Objeto Final do Programa e as organiza para conversão em linguagem de máquina. À medida que as instruções são convertidas para binário, o Montador as armazena no Objeto Final do Programa, preservando tanto o código binário quanto as informações do objeto intermediário. A Figura 3 representa a modelagem do Montador, ilustrando o fluxo de conversão das instruções para código de máquina.

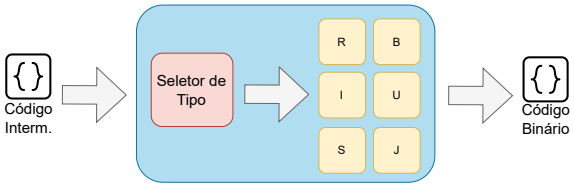

Fig. 3: *Modelagem do Montador*

C. Processador

O Processador executa o código em linguagem de máquina armazenado no Objeto Final do Programa, que contém as instruções e dados montados. Ele busca a instrução na memória com o contador de programa (PC), decodifica a instrução, identifica os registradores, a operação e os valores imediatos. A Unidade Lógica e Aritmética (ULA) realiza a operação e, se necessário, o processador acessa a memória para leitura ou escrita. Após cada execução, o estado dos registradores e da memória é atualizado, e o PC avança para a próxima instrução. O Objeto Final do Programa é atualizado para refletir o estado do sistema. A Figura 4 mostra a modelagem do processador, incluindo os componentes como memória, PC, ULA, controle e registradores.

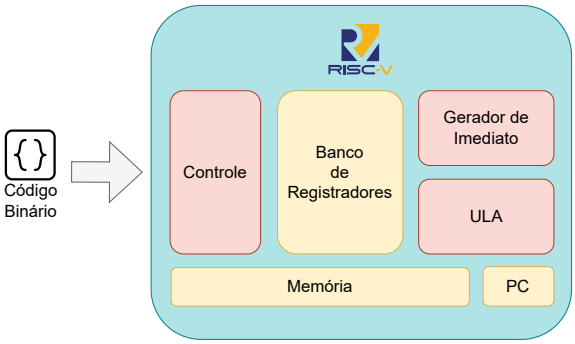

Fig. 4: *Modelagem do Processador RISC-V*

D. Objeto Final do Programa

O Objeto Final do Programa é o resultado do processo de montagem e execução, reunindo todas as informações necessárias para o simulador. Ele contém o código em linguagem de máquina, dados do programa, memórias, o contador de programa (PC) e a tabela de símbolos. O código é armazenado tanto em binário quanto na forma original, permitindo a correspondência entre o código-fonte e o executável. O objeto também inclui rótulos, posições de memória, valores dos registradores e a memória acessada durante a execução. A Figura 5 ilustra a organização dos principais componentes do Objeto Final do Programa em um diagrama de classe simplificado.

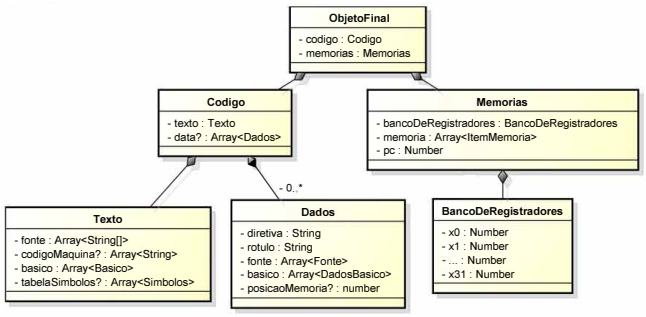

Fig. 5: *Diagrama de classes do Objeto Final do Programa*

V. RESULTADOS

Os resultados de desempenho do simulador foram obtidos usando o Chrome, versão 129.0.6668.100, para medir o consumo de memória RAM e avaliar o tamanho do núcleo.

A análise seguiu duas etapas: primeiro, mediu-se o consumo de RAM durante a execução de um teste com um código específico, usado para validar as instruções no núcleo do simulador. O consumo foi registrado com as Ferramentas de Desenvolvimento do Chrome, na aba "*Memory*". Em seguida, o tamanho do projeto foi avaliado de duas formas: verificou-se tanto o tamanho do projeto Angular, que inclui o código-fonte e outros arquivos de desenvolvimento, quanto o tamanho do build do Angular, gerado para produção, contendo apenas os arquivos otimizados para execução.

A análise foi realizada utilizando o código do algoritmo de Insertion Sort, presente no Capítulo 2 do livro Computer Organization and Design RISC-V Edition: The Hardware Software Interface [1]. Ele percorre um vetor de inteiros, ordenando os elementos por comparação e movimentação de dados entre registradores e memória. A execução desse código permitiu validar as instruções do núcleo do simulador, e durante o teste, foram capturados snapshots que indicaram um consumo aproximado de 14 MB de memória RAM.

O projeto Angular, incluindo o código-fonte e outros arquivos de desenvolvimento, possui aproximadamente 962 KB. Já o *build* do Angular, gerado para produção, foi reduzido para 172 KB. Essa diferença significativa ocorre porque o projeto é minimizado e otimizado, removendo dependências desnecessárias e compactando os arquivos JavaScript, CSS e HTML.

Por fim, o núcleo desenvolvido em Angular será utilizado em uma interface gráfica, conforme o modelo ilustrado na Figura 6. Esse núcleo será integrado à interface, gerenciando o processamento e a execução das instruções do simulador RISC-V. A interface, criada com o uso do Figma, será responsável pela interação visual com os componentes simulados, como memória e registradores.

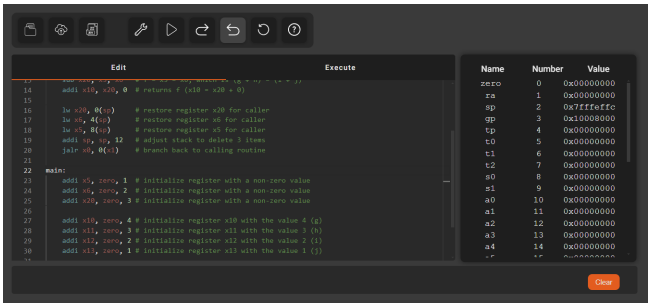

Fig. 6: *Interface do simulador*

VI. CONCLUSÃO

Neste trabalho, foi desenvolvido o núcleo de um simulador para a arquitetura RISC-V, focado no conjunto de instruções RV32I. O núcleo é composto por três componentes principais: o Analisador de Código, o Montador e o Processador, que juntos formam o Objeto Final do Programa, responsável pela execução das instruções em linguagem de máquina. O desenvolvimento utilizou TypeScript e Angular para garantir modularidade e fácil integração futura com uma interface gráfica.

O núcleo desenvolvido será integrado a uma interface web para simulação, que será disponibilizada em um repositório público, permitindo que qualquer interessado possa acessá-lo, utilizá-lo e contribuir para seu desenvolvimento. O simulador também visa ser utilizado nas disciplinas relacionadas a Arquitetura de Computadores da Universidade do Vale do Itajaí, oferecendo aos estudantes um método prático para explorar e entender a arquitetura RISC-V. Nesse contexto, como trabalho futuro, planeja-se também realizar uma avaliação do impacto do uso do simulador na curva de aprendizado dos alunos.

AGRADECIMENTOS

Este trabalho foi financiado, em parte, pela Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina – FAPESC (contratos 2023TR000880 e 2024TR001897) e pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq (processos 313513/2021-0, 350208/2022-0, 408641/2023-1 e 350794/2023-5).

REFERÊNCIAS

[1] D. Patterson and J. Hennessy, *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. The Morgan Kaufmann Series in Computer Architecture and Design, Elsevier Science, 2020.

[2] D. Robinson, "Could risc-v become a force in hpc? we talk to the experts," 2023. Acessado em: 10/10/2024.

[3] I. Scott, "Analysis on the possibility of risc-v adoption," *UC Merced Undergraduate Research Journal*, vol. 12, no. 1, 2020.

[4] C. D. Systems, "Risc-v: Democratizing innovation in cpu design." `https://community.cadence.com/cadence\_blogs\_8/b/fv/posts/risc-v-democratizing-innovation-in-cpu-design`, 2023. Acessado em: 2024-10-13.

[5] J. Nurmi, *Processor Design: System-On-Chip Computing for ASICs and FPGAs*. Springer Netherlands, 2010.

[6] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*. ISSN, Elsevier Science, 2017.

[7] M. Wolf, *Computers as Components: Principles of Embedded Computing System Design*. The Morgan Kaufmann series in computer architecture and design, Morgan Kaufmann, 2023.

[8] A. Waterman and K. Asanović, *The RISC-V Instruction Set Manual Volume I: Unprivileged ISA*, vol. 2. riscv.org, 2019.

[9] D. Patterson and A. Waterman, *The RISC-V Reader: An Open Architecture Atlas*. Strawberry Canyon LLC, 2017.

[10] A. Law, *Simulation Modeling and Analysis, Sixth Edition*. McGraw-Hill series in industrial engineering and management science, McGraw-Hill Education, 2024.

[11] R. Leupers and O. Temam, *Processor and System-on-Chip Simulation*. Springer US, 2014.

[12] K. Vollmar and P. Sanderson, "Mars: an education-oriented mips assembly language simulator," in *Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pp. 239–243, 2006.

[13] J. R. Larus, "Spim s20: A mips r2000 simulator," tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 1990.

[14] B. Landers, "Rars," 2017. Acessado em: 05/10/2024.

[15] V. M. de Morais Costa, "Vulcan." `https://github.com/vmmc2/Vulcan`, 2020. Acessado em: 07/10/2024.

[16] G. Savaton, "emulsiv." `https://eseo-tech.github.io/emulsiV/`, 2020. Acessado em: 07/10/2024.

[17] R. Agrawal, S. Bandara, A. Ehret, M. Isakov, M. Mark, and M. A. Kinsy, "The brisc-v platform: A practical teaching approach for computer architecture," in *Proceedings of the Workshop on Computer Architecture Education*, WCAE'19, (New York, NY, USA), Association for Computing Machinery, 2019.

[18] R. Giorgi and G. Mariotti, "Webrisc-v: a web-based education-oriented risc-v pipeline simulation environment," in *Proceedings of the Workshop on Computer Architecture Education*, WCAE'19, (New York, NY, USA), Association for Computing Machinery, 2019.

[19] M. B. Petersen, "Ripes: A visual computer architecture simulator," in *2021 ACM/IEEE Workshop on Computer Architecture Education (WCAE)*, pp. 1–8, 2021.