Implementações de Processadores MIPS em Simulador Visual

Christofer Rodrigues[1], **Rogério Aparecido Gonçalves**[1] **e João Fabrício Filho**[1]

[1]Departamento Acadêmico de Computação (DACOM)
Universidade Tecnológica Federal do Paraná (UTFPR) – Campus Campo Mourão
Via Rosalina M Santos, 1233 – 87301-899 – Campo Mourão – PR – Brasil

christofer_daniel12@hotmail.com,{rogerioag,joaof}@utfpr.edu.br

Abstract. *This work describes the development of a MIPS architecture, called Aperture, based on the specifications described in the book by David A. Patterson and John LeRoy Hennessy. The aim is to make the process of studying and developing components for a processor more visual and interactive, providing a ready-made basis for testing. The project was developed using* `Logisim`*, which was chosen because it is a visual simulator of digital circuits, allowing you to see the data passing through the connections or to analyze components at the logic gate level. By integrating visualization and interactivity, we aim to ease the process of creating, analyzing, and understanding a computer architecture.*

Resumo. *Esse trabalho descreve o desenvolvimento de uma arquitetura MIPS, nomeada Aperture, baseado nas especificações descritas no livro de David A. Patterson e John LeRoy Hennessy. O propósito é tornar mais visual e interativo o processo de estudo e desenvolvimento de componentes para um processador, fornecendo uma base pronta para que testes e experimentos possam ser realizados. A visualização dos dados que transitam em conexões, e a análise de componentes e portas lógicas que constituem o processador é possível por meio do uso do simulador visual de circuitos digitais* `Logisim`*. Ao integrar visualização e interatividade, visamos facilitar o processo de criação, análise e entendimento de uma arquitetura de computadores.*

1. Introdução

Por conta das próprias características físicas dos microcomputadores, é impossível visualizar o funcionamento das estruturas arquiteturais dos microprocessadores como se estivéssemos abrindo um relógio e olhando as engrenagens fazendo seu papel para o funcionamento do todo. Dessa forma, precisamos recorrer a simuladores de *hardware* para realizar testes de funcionamento, depuração, ou simplesmente para fins de aprendizado.

Trabalhos como o `EduMIPS` (Patti et al. 2012) permitem a visualização do funcionamento de uma arquitetura de computadores, mas não oferecem suporte a interações ou modificações no circuito. Isso impede um estudo minucioso sobre a implementação ou como funcionariam diferentes modificações. Essa possibilidade de inspeção do circuito possibilita melhor compreensão do que estaria acontecendo. Há então a necessidade de desenvolvimento de um ferramenta com foco na interatividade com o circuito.

O objetivo deste trabalho é implementar em um simulador visual um caminho de dados da arquitetura `MIPS`, com *pipeline* de 5 estágios, resolução de conflitos e encaminhamento. O caminho de dados funcionará como um *framework* para que desenvolvedo-

res e entusiastas possam testar implementações de diferentes componentes conhecidos na área de Arquitetura de Computadores.

Como um complemento, é disponibilizado um manual que descreve o funcionamento de cada componente usado na arquitetura, além da explicação de conceitos de lógica e circuitos digitais que podem ser úteis ao leitor.

Dessa forma, com a arquitetura completamente montada, o usuário terá em mãos não só uma visualização do circuito, mas poderá analisar as diferenças entre cada uma das etapas da evolução da arquitetura. Em cada versão disponibilizada será possível visualizar como diferentes componentes alteram o comportamento do processador. O usuário executa ou modifica cada uma delas observando como suas mudanças afetaram o funcionamento. Em outras situações, professores podem usar o material disponibilizado para explicar conceitos de arquitetura de computadores como a ideia de *pipeline* e comparar ela com uma arquitetura de ciclo único, com o bônus da capacidade de inspeção visual.

Assim, uma das principais características que diferencia este trabalho é a possibilidade do usuário modificar, retirar ou acrescentar componentes à arquitetura, realizar experimentos e observar as consequências das modificações, facilitando a análise do comportamento de um componente, ou ainda colaborando para uma compreensão mais aprofundada de como cada parte contribui para o funcionamento do todo.

2. Proposta

O estudo de Silva *et al.* (2015) constatou uma grande empolgação de alunos quando estavam participando de projetos de pesquisa na área de desenvolvimento de *hardware* com `VHDL`. Portanto, observando esse caso, e também a lacuna no estado-da-arte, em que as ferramentas existentes não oferecem a possibilidade de interação e inspeção do *hardware* desenvolvido, este trabalho implementa e documenta uma arquitetura funcional em um simulador de circuitos digitais. As implementações disponíveis até o momento são:

1. **Arquitetura de Ciclo Único (ACU)**: Essa implementação tem a característica de executar cada instrução em um único ciclo. Entretanto, já é de conhecimento que essa técnica não é a mais eficiente, pois cada ciclo de *clock* precisa durar o tempo total da propagação da instrução pelo circuito.
2. **Arquitetura com *pipeline* de cinco estágios sem resolução de conflitos (PSRC)**: Nessa arquitetura, o caminho de dados é separado em cinco estágios usando bancos de registradores intermediários, que servem para armazenar informações produzidas em um estágio anterior que serão necessárias na etapa seguinte. Nessa modificação o período de *clock* é dividido entre o número de etapas e, assim, cinco instruções podem ser carregadas simultaneamente no caminho de dados. Dessa forma, há maior *throughput* de instruções do que no ciclo único. Entretanto, alguns conflitos de dependência de dados que surgem nessa implementação precisam ser resolvidos no código com a adição de instruções `NOP` pelo *software*, que servem para dar um tempo entre uma instrução e outra, e em casos extremos, faz com que o potencial ganho de desempenho seja anulado.
3. **Arquitetura com *pipeline* de cinco estágios com resolução de conflitos (PCRC)**: Essa implementação não possui potenciais ganhos de performance sobre a anterior, mas tem o componente de resolução de conflitos integrado no processador. Isso remove o trabalho do *software* de inserir instruções `NOP` no código,

evitando potenciais problemas por falta dessas instruções, ou perdas de desempenho caso sejam colocadas instruções em excesso. A unidade de resolução de conflito garante que as instruções `NOP` sejam executadas apenas se necessário.

4. **Arquitetura com *pipeline* de cinco estágios com resolução de conflitos e encaminhamento (PCRCE)**: Essa implementação, mostrada na Figura 1, melhora o desempenho das três estratégias anteriores. Além da resolução de conflitos integrada na arquitetura, é adicionada a unidade de encaminhamento. Esse componente permite que dados sejam consumidos pela arquitetura logo quando disponíveis, evitando a necessidade de eventuais esperas entre uma instrução e outra, diminuindo significativamente a quantidade de instruções `NOP`.

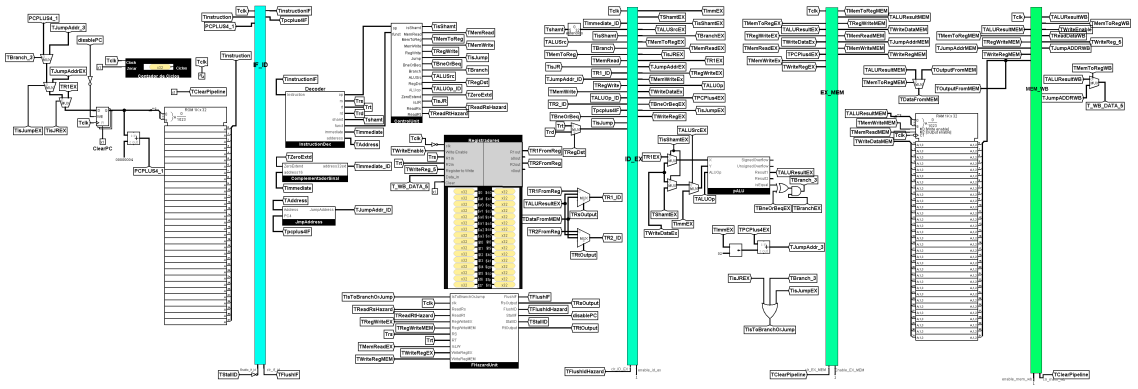

Figura 1. Versão PCRCE da arquitetura.

As implementações descritas estão disponíveis no repositório do projeto `Aperture`[1]. Futuramente, pretende-se incluir versões que implementam previsão de desvio e `VLIW`, como descrito no livro (Patterson and Hennessy 2013). A disponibilização em código aberto permite a utilização em educação e futuras pesquisas, como verificar as diferenças no funcionamento do circuito com cada um dos incrementos, e como isso afeta o número de ciclos necessários para executar um código e outras métricas como o número de instruções `NOP` que é uma métrica que poderia ajudar o usuário busca como seu código poderia ser melhorado usando reordenação de instruções.

3. Implementação

O desenvolvimento da arquitetura `Aperture` segue a descrição `MIPS` com caminho de dados de ciclo único e multi-ciclo com resolução de conflitos (Patterson and Hennessy 2013). Além disso, cada implementação está disponível isoladamente, partindo de um caminho de dados de ciclo único até a arquitetura com *pipeline* de 5 estágios com resolução de conflitos e encaminhamento.

A simulação visual é feita na ferramenta `Logisim`, que possibilita inspecionar os circuitos até o nível de abstração de portas lógicas ou visualização de sinais lógicos nas conexões entre componentes (Burch 2002). Assim, o usuário pode observar o funcionamento de qualquer item do processador enquanto executa um código que obedeça o conjunto de instruções da arquitetura. O usuário pode avançar a simulação de meio em

[1] `https://github.com/ChristoferLv/Aperture-Releases`

meio ciclo de *clock* e inspecionar os valores que estão nos fios, sinais que estão ativos ou quais valores estão armazenados nos diferentes registradores. Isso traz facilidades para futuras pesquisas, estudos e depuração do funcionamento das partes de uma arquitetura.

No desenvolvimento deste trabalho, são implementados diversos componentes da arquitetura `MIPS`, como *banco de registradores principal*, *unidade de controle*, *unidade de encaminhamento*, *bancos de registradores intermediários* e *unidade lógica e aritmética*, com algumas delas sendo adicionada de forma incremental na arquitetura. Cada componente ainda é descrito em um manual, que será um material útil para consulta e que poderá potencializar ainda mais o entendimento do usuário sobre o funcionamento da arquitetura.

4. Exemplo de Caso de Uso

Para explicar o funcionamento e verificar o desempenho das diferentes implementações, um professor poderia executar um código que calcula a soma dos valores de um vetor de dez itens, implementado em *Assembly* do `MIPS`. Com isso, o professor constataria com a turma que o número de ciclos para as implementações, seriam 31 para `ACU`, 99 para `PSRC` e `PCRC`, e 34 ciclos para `PCRCE`. Embora ACU tenha obtido esse resultado, e pareça a melhor implementação olhando somente para ciclos, seu período de *clock* tem uma duração de quase um quinto nas outras versões, o que causa uma perda potencial de quase $5x$. Dessa forma, PCRCE possui o maior desempenho potencial nessa avaliação.

5. Considerações Finais

Este trabalho possui aplicações nas áreas de educação e pesquisa em arquitetura de computadores, tornando o estudo e experimentações mais visuais e agradáveis, sendo um facilitador para exploração de entusiastas, estudantes e curiosos da área.

Ter uma opção interativa, que possibilita inspecionar o funcionamento do circuito e o benefício de poder modificar os componentes, contribui para as capacidades de entendimento em casos de estudo ou experimentos em arquiteturas, e melhora as capacidades de depuração no caso do desenvolvimento de novos componentes para um processador.

Intenções futuras para esse trabalho incluem implementação do processador usando a `ISA RISC-V` visto o interesse da indústria em arquiteturas abertas de processadores, além da implementação de despacho múltiplo de instruções.

Referências

[Burch 2002] Burch, C. (2002). Logisim: a graphical system for logic circuit design and simulation. *J. Educ. Resour. Comput.*, 2(1):5–16.

[Patterson and Hennessy 2013] Patterson, D. A. and Hennessy, J. L. (2013). *Computer Organization and Design, Fifth Edition: The Hardware/Software Interface*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition.

[Patti et al. 2012] Patti, D., Spadaccini, A., Palesi, M., Fazzino, F., and Catania, V. (2012). Supporting Undergraduate Computer Architecture Students Using a Visual MIPS64 CPU Simulator. *IEEE Transactions on Education*, 55(3):406–411.

[Silva et al. 2015] Silva, I. S., Junior, F. C. S., Patrocínio, T., and Alves, F. (2015). Aprendendo na prática: Relato de sequência de atividades práticas em iniciação científica relacionadas à arquitetura de computadores. *International Journal of Computer Architecture Education (IJCAE)*.