PS-CAS MIPS: Um simulador de pipeline do processador MIPS 32 bits para estudo de Arquitetura de Computadores

Article · January 2009

CITATION
1

READS
238

3 authors, including:

Davidson W M Nogueira
Pontifícia Universidade Católica de Minas Gerais
1 PUBLICATION **1** CITATION
SEE PROFILE

Ramon Figueiredo Pessoa
Pontifícia Universidade Católica de Minas Gerais
5 PUBLICATIONS **2** CITATIONS
SEE PROFILE

PS – CAS MIPS: Um Simulador De Pipeline Do Processador MIPS 32 Bits Para Estudo de Arquitetura de Computadores

Davidson W. N. Maia, Marcelo M. Vieira, Ramon F. Pessoa
Pontifícia Universidade Católica de Minas Gerais
{davidson.nogueira, marcelomatosvieira, ramon.fgrd}@gmail.com

Resumo

Atualmente existem diversos simuladores que auxiliam no aprendizado dos conceitos associados à técnica de pipeline*. No entanto, muitos destes simuladores demandam conhecimento avançado ainda não consolidado em alunos em estudo introdutório. Com o objetivo de permitir a melhor assimilação dos conceitos de* pipeline *por parte dos estudantes foi desenvolvido o simulador PS – CAS MIPS –* Pipeline Simulator for Computer Architecture Study *– de modo que este possa ser uma ferramenta, utilizada por professores e alunos em disciplinas de arquitetura de computadores, para propiciar uma abordagem introdutória e motivante aos conceitos de conflito por dados em* pipeline*. O simulador PS - CAS MIPS tem como principal característica a confiabilidade na execução das simulações, permitindo que o professor possa explorar conceitos relacionados à execução de instruções em um* pipeline*, de forma confiável e o aluno possa ter um ponto de referência para correção das tarefas propostas em aula ou como atividades extra-classe.*

1. Introdução

O estudo em arquitetura de computadores pode ser facilitado com a utilização de simuladores [1], [6], que permitam a abstração de detalhes. Desta forma, o aluno pode se preocupar, em um primeiro momento, com os conceitos básicos associados a *arquitetura* analisada, construindo assim, uma base de conhecimento introdutório que permitirá o avanço para tópicos mais avançados. O estudo de técnicas de pipeline, amplamente associadas às organizações arquiteturais dos processadores atuais, não foge a esta regra. Neste contexto, foram propostos e desenvolvidos diversos simuladores [2] cujo objetivo é permitir que o estudante, através de um *software* com suporte a uma interface gráfica, visualize e aprenda os conceitos estudados.

No entanto, grande parte dos simuladores analisados introduz diversas ferramentas que em muitas situações podem dificultar o aprendizado do estudante, por demandarem entendimento de conceitos relacionados à arquitetura do processador e tamanho da memória física, como pode ser verificado em [2]. Isto acarreta em desmotivação por parte do estudante em função das dificuldades encontradas. Assim, torna-se necessário o desenvolvimento de um simulador que possa ser usado como ferramenta para o estudo introdutório dos conceitos de *pipeline*, mas que mantenha a fácil operação e portabilidade.

Dentre os simuladores analisados, o simulador WebSimple MIPS [8], amplamente utilizado nas disciplinas de arquitetura de computadores, ministradas no bacharelado em Ciência da Computação, da Pontifícia Universidade Católica de Minas Gerais em Belo Horizonte, mostrou-se o mais adequado para servir de base para o desenvolvimento de um novo simulador em função de sua *fácil operação, portabilidade, código fonte acessível* e foco para o aprendizado dos conceitos de conflito por dados.

Na sequência é descrita a implementação de um simulador, baseado no simulador WebSimple MIPS, cujo objetivo do desenvolvimento é permitir a criação de um simulador pipeline de processador MIPS que resolva os possíveis conflitos por dados *sem erros*, de fácil operação e mantendo a portabilidade.

Este documento está organizado da seguinte maneira: a Seção 2 descreve os trabalhos relacionados, a Seção 3 descreve o simulador PS – CAS MIPS e faz um comparativo com o simulador WebSimple MIPS, a Seção 4 mostra alguns dos principais resultados e a Seção 5 conclui com os trabalhos futuros e contribuições.

2. Trabalhos Relacionados

A maior parte dos simuladores analisados se baseia no processador MIPS utilizado em [3]. No entanto, os níveis de detalhes com que cada implementação procura representar o *pipeline* e seus cinco estágios variam, permitindo desde simulações simples do comportamento das instruções dentro do *pipeline* até simulações em que

parâmetros de configuração da memória física podem ser fornecidos [2], [4].

Em [2], os simuladores WinDLX, MIPSim e M10KSim, são analisados e suas principais características são levantadas. O MIPSim não sinaliza de forma gráfica as mudanças de estágio dentro do pipeline. Nenhum dos simuladores acima relacionados apresentaram alguma forma de retroceder e/ou adiantar o código dentro do pipeline. Caso o aluno tenha alguma dificuldade, será necessário executar novamente todo o código para sanar essa dúvida. No que se refere a facilidade de uso, observa-se um nível de complexidade envolvida na operação destes simuladores, que pode não ser o adequado para o aprendizado das técnicas de *pipeline*, nos estágios iniciais do estudo [6], tornando estes simuladores pouco "didáticos" em um primeiro momento.

O simulador WebSimple MIPS [8] por sua vez, apresenta uma interface simples e adequada ao propósito de introdução aos conceitos de *pipeline*, pois não requer do aluno conhecimentos que vão além das noções básicas de dependências entre instruções e de resolução de conflitos por dados através de adiantamentos [3]. Além disto, a simplificação na operação do simulador permite a abstração de conceitos mais avançados, como por exemplo, organização da memória física e construção do *hardware* utilizado para a resolução de conflitos por dados. Porém, ao longo da utilização deste simulador em aulas de arquitetura de computadores, observou-se que certos conflitos por dados gerados, não foram tratados corretamente, além da não sinalização, no relatório estatístico, de algumas dependências existentes entre instruções. Estes fatores, além da motivação de inserção de novas funcionalidades culminaram na proposta e criação do simulador PS – CAS MIPS.

Desta forma, busca-se com este simulador corrigir os erros encontrados na análise de dependências e no tratamento de conflitos e implementar ferramentas que permitam ao estudante verificar a escrita do código *assembly* informado e que permitam a geração, a qualquer instante, do relatório de dependências. As soluções desenvolvidas estão descritas na próxima seção.

3. Simulador proposto e desenvolvido

Em função das necessidades descritas nas seções anteriores, o desenvolvimento do simulador "PS – CAS MIPS – Pipeline Simulator for Computer Architecture Study" (Simulador de pipeline para estudo em Arquitetura de Computadores) – se deu tomando como base o simulador WebSimple MIPS [8]. De acordo com a avaliação realizada, o simulador WebSimple MIPS é o mais indicado para este propósito uma vez que ele possui as bases necessárias para a criação de um simulador de fácil operação destinado ao estudo introdutório das técnicas de *pipeline* associadas a um processador cuja arquitetura base é a descrita em [3].

O simulador proposto e desenvolvido neste trabalho, foi construído utilizando o código fonte do simulador WebSimple MIPS, que foi disponibilizado por seus autores , escrito na linguagem de programação Java, o que contribui para a sua portabilidade, em sistemas que possuam a plataforma Java instalada. Além disto, o ambiente de desenvolvimento utilizado é o NetBeans, uma vez que o projeto inicial do simulador WebSimple MIPS foi feito utilizando-se esta mesma IDE (Integrated Development Environment). Mudanças estruturais foram feitas com a criação de uma nova unidade de controle. O software PS – CAS MIPS constitui-se como um novo simulador, não sendo apenas uma continuação do simulador WebSimple MIPS.

Com o intuito de apresentar o software PS – CAS MIPS, a Figura 1 descreve a tela inicial do simulador. Entre as implementações feitas, destaca-se o acréscimo dos botões *Assembly Analizer e Generate Report* e a inclusão de um espaço denominado *Semantic Analyzer*.

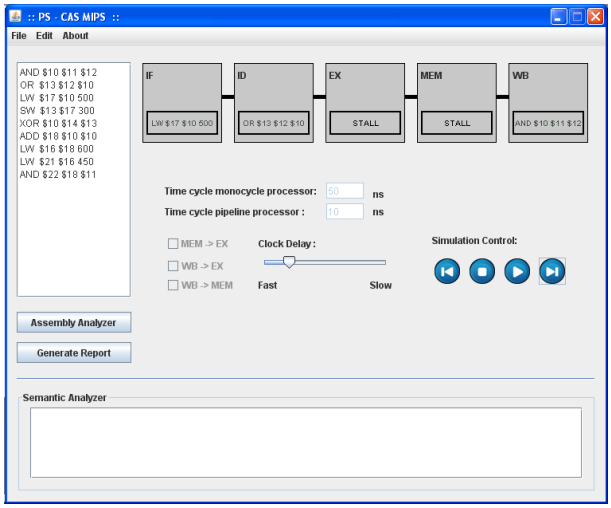

Figura 1 – Tela principal do simulador

As principais melhorias realizadas para a construção deste simulador serão descritas em detalhes a seguir.

3.1. Verificador do formato das instruções

O objetivo da análise do código *assembly* não é atuar como um analisador de sintaxe, e sim alertar o aluno que dependendo da posição do registrador na expressão, pode ocorrer uma dependência.

Esta primeira versão do analisador apenas informa ao estudante se algum tipo de operando está faltando (ou se está em excesso) para uma dada instrução, garante ainda que para as instruções `Load Word` e/ou `Store Word` exista um valor associado e cuida para que erros de sintaxe, como por exemplo, a inserção acidental de vírgulas e espaços em branco adicionais, sejam informados ao usuário. A Figura 2 mostra um erro ocorrido em função da instrução `LW $18 $10 $100` estar com a sintaxe incorreta, com a adição de um terceiro registrador, no lugar do valor `100` e da falta de um terceiro registrador para a instrução `ADD $18 $10 10`.

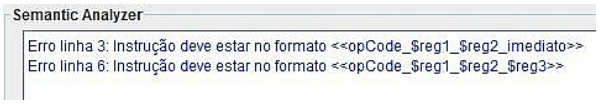

Figura 2 – Erro de código assembly

3.2. Aperfeiçoamento do código de tratamento de conflitos

Uma das principais mudanças que foram realizadas quanto ao simulador WebSimple MIPS foi a correção da forma com que certos conflitos entre instruções eram tratados durante a execução do programa. Desta forma foi construída uma "unidade de controle" inteiramente nova para possibilitar a análise adequada destes conflitos, levando-se em conta quando os adiantamentos estavam ativados. Embora esta modificação não esteja visível ao usuário, pois se trata de uma alteração em código fonte na parte estrutural propriamente dita, ela é um ponto de partida para que novas funcionalidades sejam inseridas e contribui para a confiabilidade da simulação. A Figura 3 mostra os três adiantamentos possíveis para a resolução de conflitos por dados, disponíveis no simulador, sendo eles *Memória de dados → Execução (MEM-EX), Escrita de resultado → Execução (WB-EX), Escrita de resultado → Memória de dados (WB-MEM).*

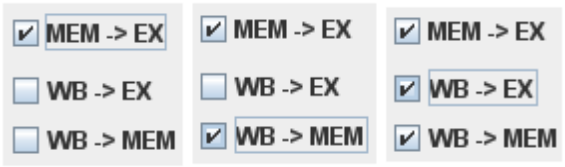

Figura 3 – Alguns Adiantamentos do simulador

3.3. Geração assíncrona de relatório estatístico

No simulador PS – CAS MIPS torna-se possível gerar um relatório estatístico em qualquer instante posterior à análise do código *assembly* informado. O estudante pode optar por gerar o relatório antes do término da execução: neste caso, o relatório exibido mostrará apenas as dependências existentes entre as instruções (mesmo que não gerem potenciais conflitos). Caso contrário, o aluno pode terminar a simulação e o relatório final será exibido automaticamente, contendo não somente as dependências, mas também, os dados relacionados à quantidade de bolhas inseridas no pipeline para a resolução dos eventuais conflitos por dados e os dados relacionados ao *speed up* [3] proporcionado pela utilização do *pipeline* se comparado com um processador monociclo. A Figura 4 mostra um relatório completo, gerado depois de uma simulação.

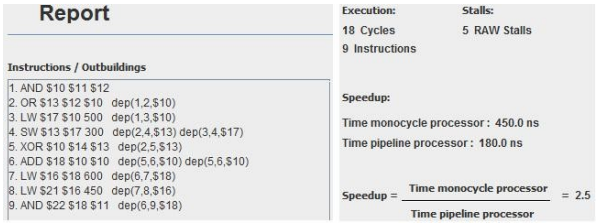

Figura 4 – Exemplo de relatório gerado

4. Resultados

A flexibilidade mostrada na utilização do simulador PS – CAS MIPS contribuiu para a execução rápida de inúmeros conjuntos de instruções, de modo que foi possível verificar em cada caso, todas as dependências associadas e o número de bolhas geradas em função da existência ou não de adiantamentos ativados. Além disto, foram realizados testes relacionados à capacidade de navegação pelo simulador, isto é, a possibilidade de avançar e retroceder, passo-a-passo na execução de instruções, o que contribui para o aprendizado do estudante que esteja utilizando o simulador PS – CAS MIPS em seus estudos, orientados pelo professor.

A busca pela criação de um simulador de pipeline de processador MIPS que estivesse sem erros, fez com que os desenvolvedores do PS – CAS MIPS realizassem inúmeros testes em seu simulador, o que contribuiu para que situações onde erros foram observados no WebSimple MIPS não mais existissem no PS – CAS MIPS, proporcionando uma maior confiabilidade do simulador.

No Gráfico 1 é observado que algumas dependências para um, dois e três adiantamentos antes não tratados corretamente pelo simulador WebSimple MIPS, já não se apresentam mais como um problema no simulador PS – CAS MIPS, garantindo uma maior confiabilidade e eficácia do novo simulador proposto e implementado. A Tabela 1 foi utilizada como base para a criação do Gráfico 1.

ADIANTAMENTOS ATIVADOS		
WB -> EX	MEM -> EX e WB -> MEM	TODOS
XOR $20, $5, $11	ADD $5, $9, $3	AND $10, $15, $30
AND $12, $11, $12	SUB $12, $5, $8	ADD $12, $10, $18
ADD $22, $20, $26	LW $13, $12, 6	LW $5, $10, 30
LW $52, $12, $200	SW $13, $7, 58	SW $5, $13, 50

Tabela 1 – Códigos Assembly utilizados na criação do Gráfico 1

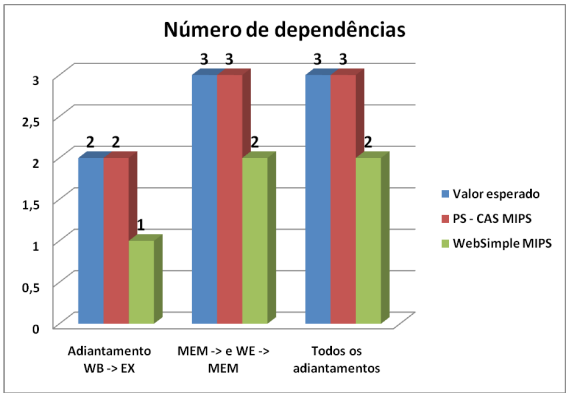

Gráfico 1 – Dependências para um, dois e três adiantamentos ativados

A execução dos códigos assembly da Tabela 1 foram realizados na versão do simulador WebSimple MIPS disponível em http://matheus.ath.cx/simple/?page=downloads até o dia 31/07/2009. O número de bolhas para os três códigos da Tabela 1 também não estavam corretos na versão do WebSimple MIPS disponível no inicio do ano de 2009, no entanto, com a disponibilização de uma versão atualizada do simulador feita pelos seus desenvolvedores, o problema foi resolvido. O relatório para o código com todos os adiantamentos ativados gerados no PS – CAS MIPS e WebSimple MIPS estão disponíveis na Figura 5.

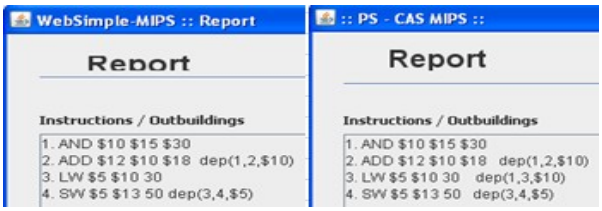

Figura 5 – Relatórios para a 3ª sequência de códigos

5. Conclusão e contribuições

Este trabalho apresentou o simulador PS – CAS MIPS – *Pipeline Simulator for Computer Architecture Study* que mostrou ser em seus testes um bom sucessor do simulador WebSimple MIPS, em função da confiabilidade atingida durante a simulação da execução das instruções. Este simulador contribui com o estado da arte, de modo que torna-se uma alternativa portável e de fácil operação para o estudo introdutório dos conceitos relacionados à técnica de *pipeline*.

Como trabalhos futuros, pretende-se implementar as soluções de conflitos por controle provocados por desvios condicionais, a possibilidade de simulação de um pipeline com escalonamento dinâmico, recursos gráficos que sinalizam, através de cores, o instante em que um adiantamento acontece, e o aperfeiçoamento do verificador de instruções.

6. Referências

[1] BRORSSON, M. MipsIt: a simulation and development environment using animation for computer architecture education. In Proceedings of 2002 Workshop on Computer Architecture Education: Held in Conjunction with the 29th international Symposium on Computer Architecture (Anchorage, ACM, New York, NY, 12. p. WCAE'02.Alaska),p.1 -8.

[2] GRUNBACHER, H., Teaching computer architecture/organisation using simulators, 28th Annual Frontiers in Education Conference, p.1107-1112 vol. 3, 1998.

[3] HENNESSY, J., PATTERSON, D, Organização e Projeto de Computadores, 3ª edição. Ed. Compus 2005.

[4] KOSHRAVIPOUR, M. WinDLX disponível em http://cs.uns.edu.ar/~jechaiz/arquitectura/windlx/windlx.html, acessado em junho de 2009.

[5] MARTINS, C.A.P.S.; RAMOS, L.E.S.;CORREA, J.B.T.; GOES,L.F.W.; MEDEIROS, T.H.; A New learning method of microprocessor architecture in Frontiers in Education, 2002. FIE 2002, 32nd Annual, Volume 3, 6-9 Nov. 2002, Page S1F 16 - 21 .

[6] RODRIGUES, R.P., MARTINS, C.A.P.S. Ensino e aprendizado de pipeline de modo motivante e eficiente utilizando simuladores didáticos. Workshop Sobre Educação em Arquitetura de Computadores – WEAC, 2008, p. 25 - 28.

[7] SCOTT, M.WinMIPS64, Disponível em:http://www.computing.dcu.ie/~mike/winmips64.html . Acessado em Junho /2009

[8] SOUZA, B.F. de, MOREIRA, M.P.A., NOGUEIRA R. S., MARTINS C.A.P.S. WebSimple MIPS. Workshop de Sistemas Computacionais de Alto Desempenho – WSCAD, 2008, p. 1 – 4.