PASAC: Uma ferramenta de auxílio no estudo da execução de *Pipeline* em processadores.

Carlos F. Bublitz[1], Diego Pittol[1], Eduardo F. Brancher[1], Leonel P. Tedesco[1]

[1]Universidade de Santa Cruz do Sul (UNISC)
96.815-900 – Santa Cruz do Sul – RS – Brasil

carlosbublitz@mx2.unisc.br, diegopittol@mx2.unisc.br,
ebrancher@mx2.unisc.br,leoneltedesco@unisc.br

Abstract. *This paper describes the PASAC, a tool to support the teaching of computers architecture, capable of simulating a Pipeline data path, accepting MIPS Assembly as input code. The article demonstrates the main features of software operation, bringing relevant topics in more details such as the identification and treatment of Hazards and visualization of the activation of modules in each Pipeline stage.*

Resumo. *Este artigo descreve o PASAC, uma ferramenta de apoio ao ensino de arquitetura de computadores, capaz de simular um caminho de dados Pipeline, aceitando como entrada códigos em Assembly MIPS. O artigo demonstra as principais características de funcionamento do software, trazendo com mais detalhamento a identificação e tratamento de Hazards e visualização da ativação dos módulos em cada estágio do Pipeline.*

1. Introdução

Com a crescente necessidade de maior desempenho em sistemas computacionais, instalou-se uma busca incessante por novas técnicas, abordagens e métodos, como as arquiteturas multicores, onde vários núcleos são inseridos dentro de um encapsulamento, proporcionando um alto grau de paralelismo na execução de tarefas. A técnica *pipelining* é um fator determinante quando se fala em desempenho, fazendo assim necessário a busca pelo domínio desta técnica no estudo das arquiteturas computacionais atuais. A difícil aprendizagem da técnica *pipelining* em um ambiente exclusivamente teórico traz à tona a necessidade de ferramentas auxiliares capazes de simular o funcionamento de uma arquitetura *pipeline* de forma visual, exemplificando os conceitos teóricos vistos pelo aluno, pois, de acordo com Cassel [1], a simulação permite que se verifique o funcionamento de algum sistema real em um ambiente virtual, gerando modelos que se comportam como aquele, considerando a variabilidade do sistema e demonstrando o que acontecerá na realidade de forma dinâmica.

Esta ferramenta de simulação foi desenvolvida para preencher esta lacuna, agregando valor ao processo de aprendizagem, sendo útil para usuários com níveis variados de conhecimento, desde iniciantes até usuários de nível avançado. Seu desenvolvimento deu-se no contexto da disciplina de Arquitetura de Computadores da UNISC, com o intuito de criar uma ferramenta de apoio ao ensino que atendesse a certos requisitos, focando em pontos de difícil compreensão da matéria.

2. PASAC (Pipeline Architecture Simulator with Assembly Codes)

Observando a necessidade da existência de uma ferramenta confiável, que proporcione ao usuário a possibilidade de executar códigos em Assembly MIPS, visualizar os estágios do *pipeline* e a detecção de dependências, tanto de dados como de controle, tendo como diferencial a exibição em

hardware do *pipeline* e suas unidades, algo não comumente encontrado nos simuladores, iniciou-se o desenvolvimento do PASAC: um simulador que possui as características desejadas, acrescentando funções simples, porém de importância significativas, na operabilidade do programa, tornando-o agradável ao usuário, como a opção de abrir e salvar códigos, determinar velocidade para a simulação e a detecção de erros nos Códigos Assembly, fornecendo ao usuário *feedback* necessário para que o mesmo compreenda o erro existente e possa corrigi-lo.

Optou-se pelo uso da linguagem Java para o desenvolvimento da ferramenta, uma vez que esta possui características importantes de compatibilidade com diversas plataformas. Para a implementação, foi usada a IDE NetBeans [4], já de conhecimento e domínio dos desenvolvedores, proporcionando produtividade na criação da interface visual através de seu sistema de "arrastar e soltar". O PASAC tem seu pipeline baseado no modelo de cinco estágios proposto em [5].

Buscando criar um ambiente familiar e agradável ao usuário, usou-se de artefatos como abas, separando o código, a simulação e o relatório, assim como também separando banco de registradores e memória principal. Uma visão geral do programa encontra-se na figura número 1.

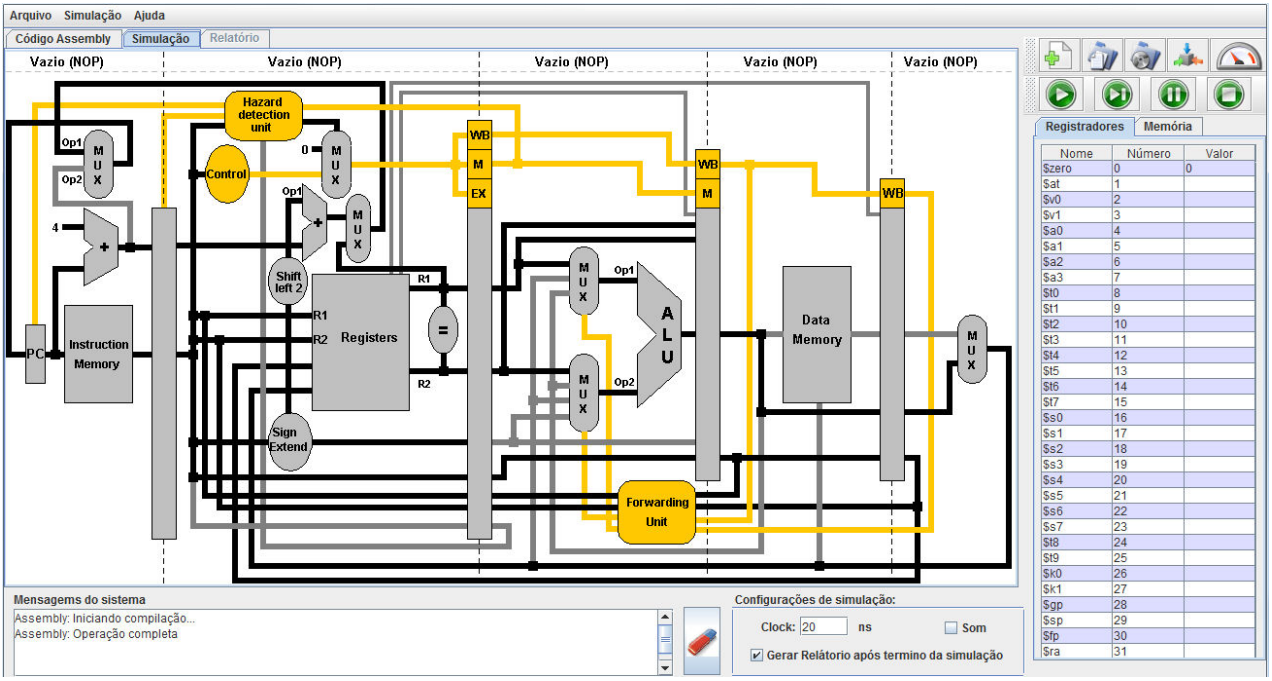

Figura 1. Tela do programa completa, exibindo a aba Simulação e a aba Registradores.

2.1 Exibição de *Pipeline*

Um dos importantes diferenciais da Ferramenta é a exibição de um *pipeline* completo, com seus módulos e caminhos de dados, conforme visto na figura 2, facilitando a compreensão do usuário a cada passo executado. Com o *pipeline* cheio, o modelo fica um pouco poluído, resultado da complexidade da execução da técnica. O usuário, porém, pode fazer uso do controle de velocidade de execução, diminuindo a velocidade para ter mais tempo de analise, ou até, caso prefira, executar o programa ciclo a ciclo, usando um botão especifico para isso. A opção de executar um ciclo de cada vez torna a ferramenta útil para usuários que queiram analisar detalhadamente como o hardware de um *pipeline* se comporta.

As linhas de dados do modelo de hardware da ferramenta se colorem conforme são ativadas, voltando novamente à sua cor original quando não forem mais necessárias, conforme a figura 3, evidenciando assim o percurso utilizado pela instrução durante sua execução. A cor das linhas enquanto ativadas é vermelha, e, quando não mais utilizadas, voltam a suas cores originais: amarelo para sinais de controle, preto para linhas principais e cinza para linhas secundárias.

São retratadas no modelo todas as unidades de importância para o programa, inclusive as unidades de dependência de dados, que são descritas a seguir, porém, a fim de evitar que o modelo se torne incompreensível, apenas os sinais de controle relativos às dependências de dados são exibidos, deixando de lado a exibição dos sinais de controle padrão, ficando subentendido que o comportamento das unidades é controlado por sinais.

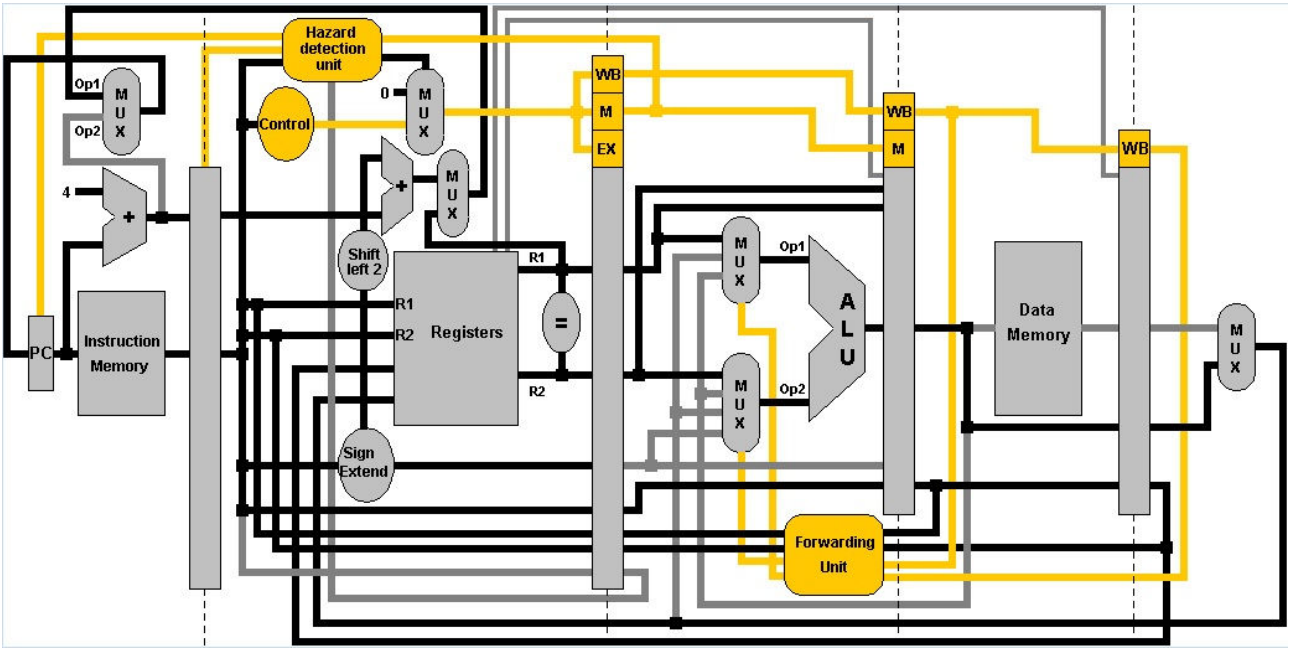

Figura 2. Modelo completo do hardware exibido na ferramenta.

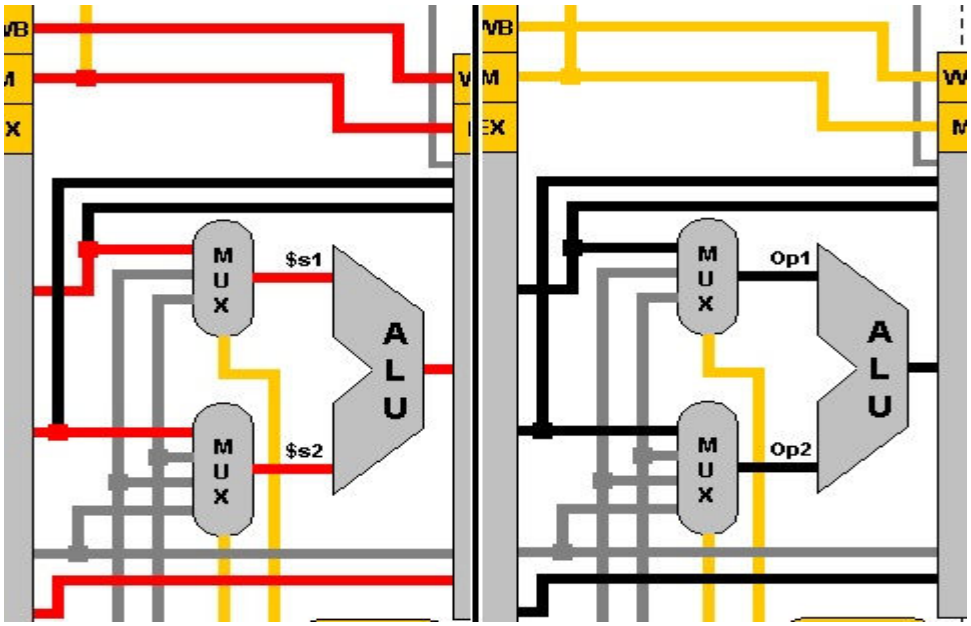

Figura 3. À esquerda linhas coloridas durante utilização, e à direita em seu estado original.

2.2 Tratamento de Dependências

Devido à complexidade de sua estrutura, durante a técnica *pipelining* algumas dependências são geradas. Essas dependências podem ser tratadas de maneiras distintas. Na sequência, os métodos usados pela ferramenta para tratar estas dependências são explicados.

As dependências de controle são geradas quando o pipeline se depara com uma instrução de salto, que é responsável por criar laços de repetição e desvios. As instruções de salto fazem com que a continuidade do programa seja alterada, e, por este motivo, a instrução que se encontra

no primeiro estágio pode não ser a correta, tendo o hardware que buscar novamente a instrução. Para tratar esta dependência, faz-se uso da inserção de bolhas, que são instruções vazias que apenas atravessam o *pipeline*, estágio a estágio, sem executar ação alguma. A introdução destas bolhas é controlada pela *Hazard Detection Unit*, que envia sinais de controle ao primeiro estágio, responsável pela busca da instrução, o informando qual o endereço da próxima instrução a ser buscada, e tornando a instrução já buscada em bolha, realizando assim um salto. Todo este processo pode ser acompanhado pelo usuário, conforme figura número 4.

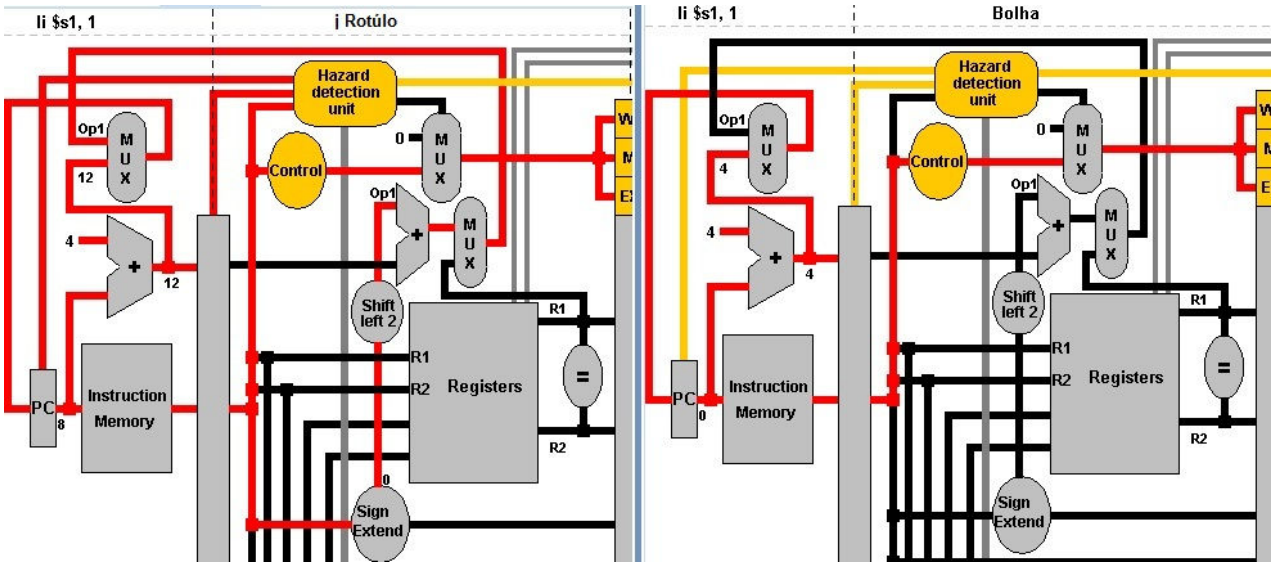

Figura 4. Primeiro momento, unidade de *Hazard* ativada, paralisando pc e informando divisória sobre a bolha. No segundo momento, a bolha já inserida, e a instrução correta entrando no *pipeline*.

As dependências de dados são geradas quando um dado é compartilhado por mais de uma instrução presente no pipeline. Este dado só será gravado e atualizado no banco de registradores no quinto estágio, fazendo com que, caso uma instrução que tenha o alterado ainda não tenha chegado a este estagio, e outra instrução que dependa dele o busque, receberá um dado incorreto. Para evitar que isto ocorra, são criados atalhos, que passam o dado de um estágio anterior diretamente para onde ele está sendo solicitado. Estes atalhos são gerenciados pela *Forwarding Unit*, que analisa as instruções, identificando as dependências e adiantando o dado do estágio em que ele se encontra para o estágio em que ele é solicitado. Todo este processo pode ser acompanhado pelo usuário, conforme figura número 5. Ainda existe a dependência de dados para instruções de *Load*, estas instruções são responsáveis por ler dados na memória principal e escrevê-los em registradores, porém este dado só estará disponível no quinto estágio. Não sendo possível adiantá-lo para resolver esta dependência, são inseridas bolhas, que congelam o pc e mantêm as instruções esperando até que o dado esteja disponível, para só então liberar seu andamento. O processo de identificação e inserção de bolhas é realizado pela *Hazard Detection Unit*.

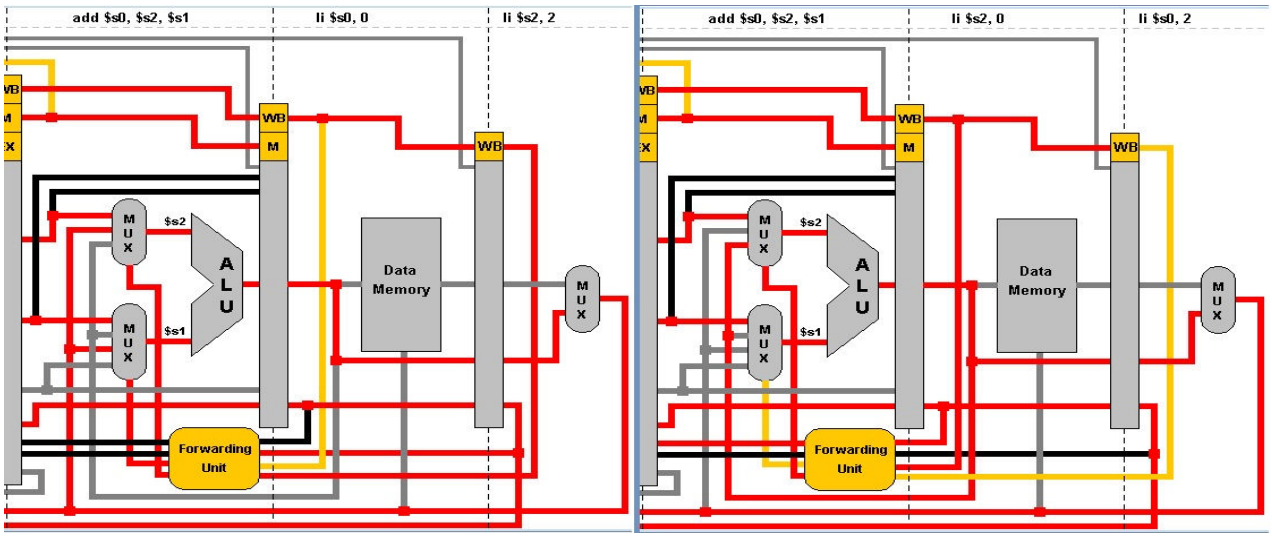

Figura 5. Primeiro momento *forwarding wrb/exe*, segundo momento *forwarding mem/exe*.

2.3 Instruções

A ferramenta conta com uma área de texto destinada a inserção de instruções em Assembly, possibilitando ao usuário inserir manualmente as instruções ou abrir um arquivo salvo em seu computador com o código desejado. Após inserção das instruções, elas poderão ser salvas em um arquivo para consulta posterior.

São aceitas instruções de salto, de registradores e de imediato, totalizando vinte e três instruções, possibilitando assim uma maior flexibilidade no desenvolvimento de códigos a serem simulados. As instruções aceitas pela ferramenta podem ser vistas na tabela número 1.

Tabela 1. Instruções aceitas pela ferramenta.

Tipo-R	Tipo-I	Tipo-J
ADD	ADDIU	BEQ
SUB	SUBIU	BNE
LW	LI	J
SW	SLL	JAL
SLT	SRL	JR
MOVE	LUI	
AND	ANDI	
OR	ORI	
XOR	XORI	

Cada instrução é simulada pela ferramenta e seus resultados são armazenados em tempo de execução em um banco de registradores, permitindo que o usuário acompanhe todas as alterações feitas no banco de registradores pelas funções, compreendendo também em que estágio esta alteração ocorrerá. Também é possível que o usuário acompanhe o estado da memória principal, já que, assim como o banco de registradores, sua atualização ocorre em tempo de execução. Desta forma, o usuário compreende em qual estágio uma instrução de *Load* grava na memória, além de poder acompanhar quais dados se encontram nela. As tabelas de visualização do banco de dados e

da memória principal podem ser vistas na figura número 6.

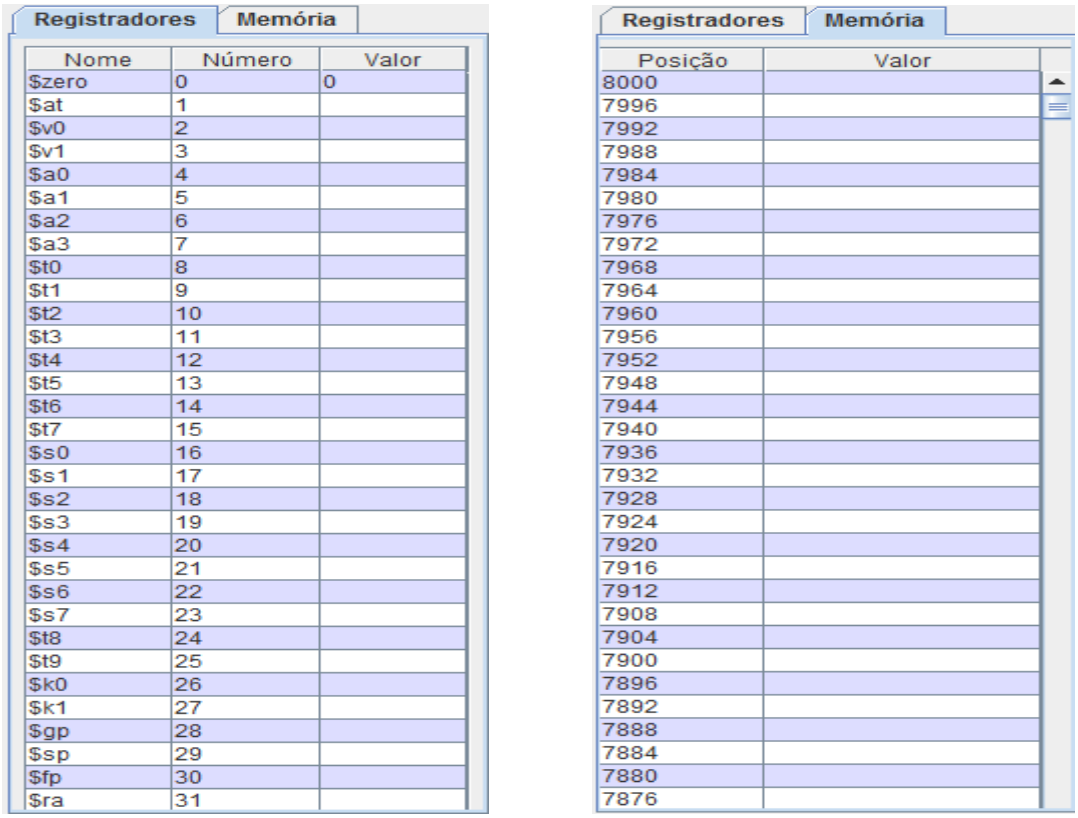

Figura 6. Banco de registradores à esquerda e memória principal à direita, sendo sua visualização na ferramenta alternada através de abas.

Além das instruções em Assembly MIPS, é possível criar rótulos que são utilizados posteriormente pelas instruções de salto.

A sintaxe e funcionalidade de todas as instruções aceitas pela ferramenta podem ser consultadas através do menu “Ajuda” localizado na barra de menus, trazendo ao usuário um importante guia para consulta de sintaxe de cada instrução.

2.4 Emissão de relatório

Para auxiliar na comparação de diferentes códigos Assembly MIPS em máquinas com e sem pipeline, ao fim de cada simulação é gerado um relatório com informações como: *clock*, números de ciclos, tempo de execução, número de instruções, número de bolhas e número de dependências nos diferentes estágios do pipeline, código que foi executado, além do status da execução, informando o usuário se ela foi interrompida por erros ou completada com sucesso.

A comparação entre o número de ciclos e tempo de execução em máquinas com pipeline e sem pipeline são de grande importância para demonstrar o ganho no desempenho em sistemas que usam essa técnica, tornando visível ao usuário o ganho proporcionado. Este relatório é um importante diferencial da ferramenta, pois não se encontra disponível em outras ferramentas como [2] e [3]. Um modelo do relatório exibido pode ser vista na figura número 10.

O *clock* do sistema pode ser alterado conforme o usuário desejar, permitindo uma maior flexibilidade e aproximação ao sistema que se deseja simular. No painel responsável pela modificação do *clock* também é possível desativar a geração de relatórios ao final das simulações, tornando opcional ao usuário sua visualização. Este painel pode ser visto na figura número 11.

Após a emissão do relatório, o usuário pode exportá-lo para um arquivo texto, caso queira realizar consultas posteriores, tornando-o uma importante fonte para comparação, sendo assim

possível uma busca por um melhor desempenho ao analisar-se diferentes resultados, ficando a cargo do usuário decidir se deseja ou não guardá-lo.

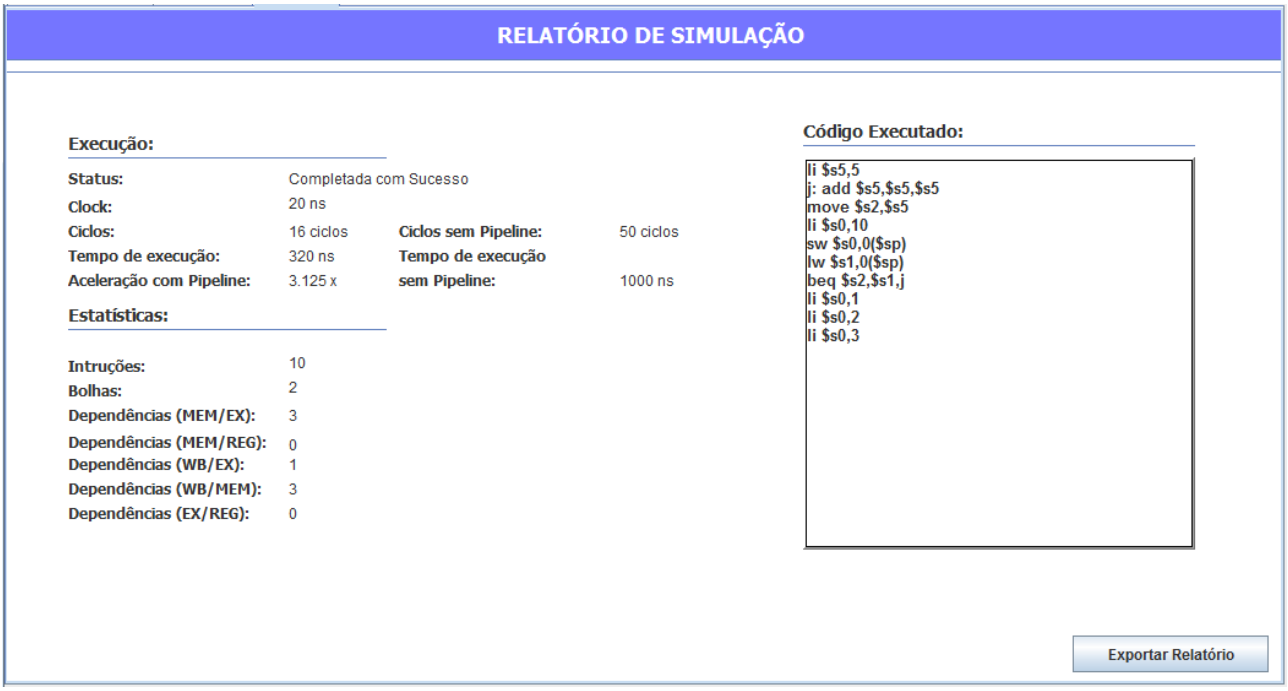

Figura 10. Relatório de simulação da ferramenta.

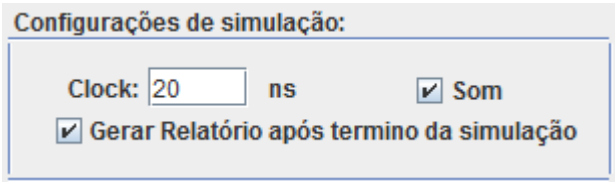

Figura 11. Painel responsável pela escolha do tempo de *clock*, geração de relatórios e ativação da funcionalidade para emissão de sons.

3. Trabalhos relacionados

Analisamos outras duas ferramentas de simulação voltadas para a disciplina de Arquitetura de Computadores, procurando mostrar suas principais características para posterior comparação com o PASAC.

O PS – CAS Mips [3] é uma ferramenta desenvolvida para o auxilio de alunos e professores na disciplina de Arquitetura de Computadores. Implementado com base em um simulador já existente (WebSimple MIPS [7]), o PS-CAS Mips apresenta importantes melhorias no que diz respeito ao tratamento de dependências e novas funcionalidades.

Possuindo uma interface simples e de fácil utilização, conta com uma área para a inserção das instruções assembly, na qual sua simulação acontece em um pipeline de cinco estágios. Durante a simulação são exibidas as instruções de cada estágio, abstraindo alguns conceitos mais complexos, como as linhas de controle e status do banco de registradores e memória, o que impossibilita a validação dos valores que as instruções deveriam gerar.

É possível escolher quais os tipos de adiantamento de dados (MEM->EX, WB->EXE, WB->MEM) podem ser usados durante a simulação, permitindo que seja feita uma comparação entre números de ciclos com ou sem o adiantamento.

Como também acontece no PASAC ao término de cada simulação, é possível gerar um relatório com algumas informações básicas como o número de ciclos, bolhas e as instruções

executadas.

Outro trabalho semelhante é o Ultimate Pipeline Simulator [8], que exibe em tempo real o número de ciclos, bolhas e a instrução atual. Tanto o Ultimate Pipeline Simulator como o PASAC tem a mesma característica de sinalizar graficamente quando ocorre uma dependência de dados e controle.

Os dois simuladores analisados implementam um número pequeno de instruções e não aceitam a criação de rótulos e nem instruções de salto.

A exibição e atualização em tempo real do banco de registradores e memória são um dos principais diferenciais do PASAC, permitindo que o aluno acompanhe os resultados gerados pelas instruções inseridas.

As principais características de cada simulador podem ser comparadas na tabela 2.

Tabela 2. Comparação das características entre os simuladores.

Característica	PS-CAS Mips	Ultimate Pipeline Simulator	PASAC
Visualização do banco de registradores	NÃO	NÃO	SIM
Visualização da memória	NÃO	NÃO	SIM
Visualização das dependências de dados	NÃO	SIM	SIM
Visualização das linhas de controle	NÃO	NÃO	SIM
Execução passo a passo	SIM	SIM	SIM
Opção de retroceder simulação	SIM	SIM	NÃO
Escolher recursos de adiantamento de dados	SIM	SIM	NÃO
Relatório ao termino de execução	SIM	SIM	SIM
Som ao detectar bolha	NÃO	SIM	SIM
Identificação de erros de compilação	SIM	SIM	SIM
Instruções de Salto	NÃO	NÃO	SIM

4. Conclusão

Com um ambiente simples e intuitivo, o PASAC é capaz de auxiliar no aprendizado da técnica de *pipeline* tanto a usuários avançados como iniciantes. Com a ferramenta, é possível visualizar a simulação das instruções, inserção de bolhas, detecção de *hazards* de dados e de controle, estado do banco de registradores e da memória principal, além de um completo relatório final de simulação. Todas essas funcionalidades tornam essa ferramenta uma das mais completas para a simulação de *pipeline*.

O uso do PASAC para o estudo da técnica de pipeline se mostra importante no processo de aprendizado, permitindo que o usuário veja na prática o comportamento do *pipeline* em diferentes situações de simulação, podendo comparar resultados que mostram o ganho de desempenho gerado pela técnica.

O PASAC passará a ser utilizada nas disciplinas de ensino de Arquitetura de Computadores nos cursos do Departamento de Informática da UNISC, tornando-se aliado do corpo docente e possibilitando uma avaliação de sua influencia na aprendizagem do aluno.

Como trabalhos futuros, pretende-se implementar a opção de escolha de cor para cada estágio, suporte a multi-idioma, geração de gráficos para análises de desempenho, entre outras, com o intuito de tornar a ferramenta ainda mais completa.

Referências

[1] CASSEL, R.A., Desenvolvimento de uma abordagem para a divulgação da simulação no setor calçadista gaúcho. Dissertação de Mestrado apresentada ao Programa de Pós-Gradução em Engenharia de Produção (PPGEP) da Universidade Federal do Rio Grande do Sul (UFRGS), 1996.

[2] MARS. Disponível em <http://www.cs.missouristate.edu/MARS/ >. Acesso em 21 de fevereiro de 2012.

[3] ProcSim MIPS. Disponível em < http://jamesgart.com/procsim/>. Acesso em 21 de fevereiro de 2012.

[4] Framework NetBeans. Disponível em <http://www.netbeans.org>. Acesso em 21 de fevereiro de 2012.

[5] D. A. Patterson and J. L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, 3nd ed., Morgan Kaufmann Publishers, 2007.

[6] PS – CAS Mips: Simulador de Pipeline do Processador MIPS 32 bits para o estudo de Arquitetura de computadores. Disponível em: <https://sites.google.com/site/pscasmips/home>. Acesso em 21 de fevereiro de 2012.

[7] Websimple - MIPS: Simulador web-based do pipeline do MIPS. Disponível em: <http://matheus.ath.cx/simple/>. Acesso em 21 de fevereiro de 2012.

[8] Ultimate Pipeline Simulator: Simulador do Pipeline do MIPS. Disponível em: < www.inplanet.com.br/ups>. Acesso em 21 de fevereiro de 2012.