Entendendo as Relações entre Compilador, Arquitetura e Organização com o Simulador do MIPS I Gerado por um Framework de Modelagem

Daniel Westerlund, Matheus Antonio Zucolotto Pereira, Sandro Neves Soares

Campus Universitário da Região dos Vinhedos – Universidade de Caxias do Sul – Bento Gonçalves – RS – Brasil

dwesterl@ucs.br, mazperei@ucs.br, snsoares@ucs.br

Resumo. *Este trabalho apresenta um simulador do processador MIPS I, cujos acréscimos sobre outros simuladores de mesmo propósito são: o fato de ele ser gerado automaticamente por um framework de modelagem; e a possibilidade de usar, como entradas, programas escritos na linguagem C. Ele propicia ao estudante uma visão completa do processo de execução de um programa, desde a sua criação usando uma linguagem de alto nível até a sua execução passo a passo pelos componentes do hardware. O estudante pode, entre outras facilidades, visualizar e entender como se dá a passagem de argumentos para uma função usando o segmento de pilha do programa.*

Abstract. *This work presents a graphical simulator of the MIPS I processor, whose advantages over the other simulators employed in education are (1) it is created automatically by a design framework, and (2) the possibility of using C programs as its inputs. The simulator provides a complete view, to the student, about the program execution process, from the source code creation, using a high level programming language, to the code execution, step by step, by the processor datapath components. The student can visualize and comprehend, for example, how the program passes arguments to a function using the stack segment.*

1. Introdução

Este trabalho apresenta um simulador do processador MIPS I, que permite a estudantes visualizar graficamente as microoperações e o estado dos componentes da organização do processador ao executar programas Assembly, gerados a partir de programas escritos na linguagem C e compilados com o GNU gcc.

Os acréscimos desta ferramenta sobre outras de mesmo propósito, como SPIM [Laurus 2004], DARC2 [Uy 2004] e ESCAPE [Verplaetse, Campenhout and Neers 1999], é o fato de ela ser gerada automaticamente por um *framework* de modelagem [Soares and Wagner 2006] (simuladores de outros processadores estão disponíveis igualmente) e a possibilidade de usar programas em C como entradas para o simulador. Esta última característica propicia ao estudante uma visão completa do processo de execução de um programa, desde a sua criação usando uma linguagem de alto nível até a sua execução passo a passo pelos componentes do hardware, passando pelo seu carregamento na memória do processador. O estudante pode visualizar o código Assembly (e o código binário) gerado a partir do seu programa em C, acompanhar como

se dá a passagem de argumentos para uma função e as diferenças no armazenamento de variáveis locais e globais, entre outras facilidades.

A seção que segue apresenta como usar o simulador e a última seção, por sua vez, lista alguns resultados já obtidos.

2. Visão Geral

A simulação de um programa na ferramenta se dá com a carga, pelo usuário, de um arquivo de texto com instruções em Assembly. Este arquivo pode ser escrito usando a linguagem Assembly diretamente, ou pode ser gerado a partir de um programa escrito em C e, após, traduzido para Assembly pelo compilador GNU gcc. O arquivo em Assembly é traduzido pelo *Montador do MIPS I*, integrado à ferramenta, para código binário. Este código binário é, então, carregado nas respectivas memórias de instruções e de dados. A execução do programa pode ser feita ciclo a ciclo, por múltiplos ciclos ou até o seu final. Ao término da simulação, o usuário pode visualizar os resultados na própria GUI (*Graphical User Interface*) do simulador (ver janela da direita na Figura 1).

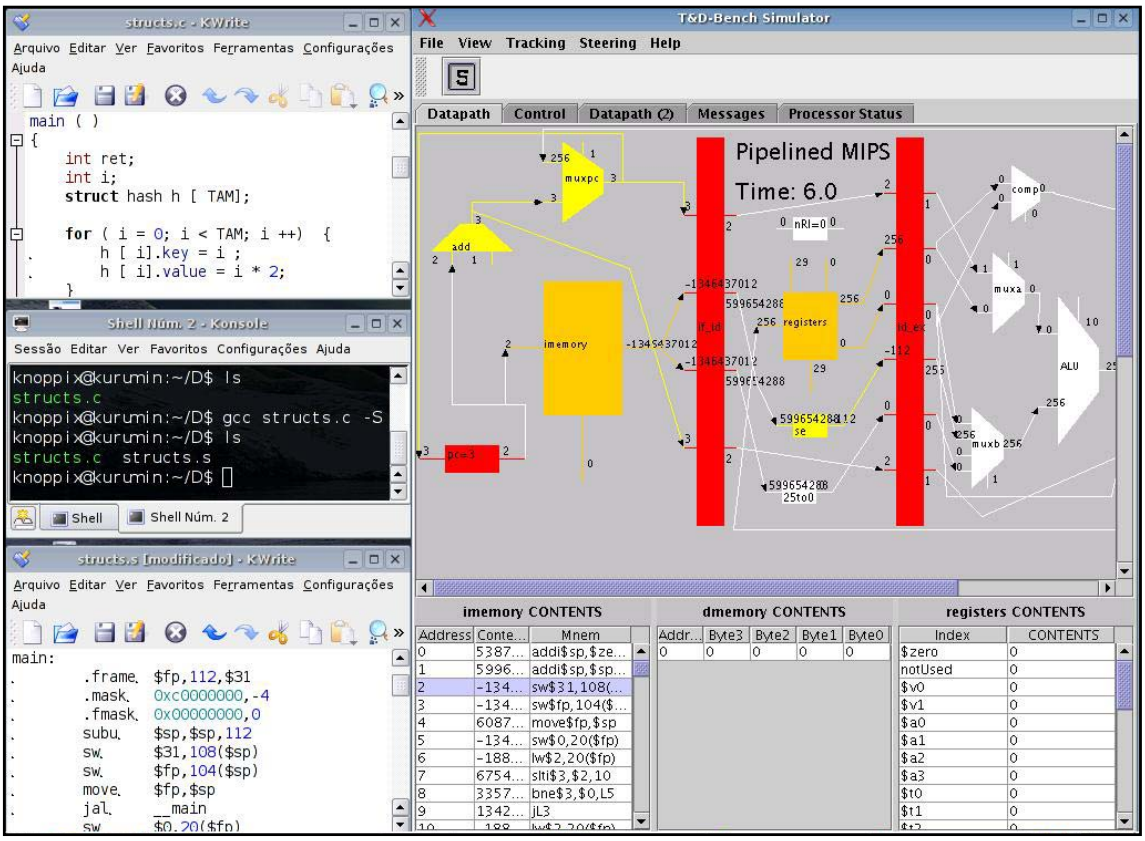

Figura 1. Compilação de código C para Assembly e Janela Principal do Simulador.

A Figura 1 ilustra o processo de criação de um programa em C a ser executado no simulador. A janela posicionada na parte superior à esquerda mostra o programa em C. A janela ao centro, no lado esquerdo, mostra o uso do compilador GNU gcc, configurado para gerar código MIPS. A janela posicionada no canto inferior esquerdo mostra o respectivo programa em Assembly gerado, e a janela posicionada no lado direito é a janela principal do simulador do processador MIPS I, onde se pode ver os conteúdos das memórias de dados, de instruções e do banco de registradores, além do

diagrama de blocos que mostra os vários elementos da organização do processador. Diversas outras informações são disponibilizadas nas demais abas da janela principal do simulador, mas não estão sendo mostradas na figura. Dentre elas, podemos citar: o *pipeline* do processador com as respectivas instruções em execução (onde se pode ver, por exemplo, o uso de bolhas para o tratamento de instruções com dependências de dados), os detalhes destas instruções (como valores nos seus campos de bits) e mensagens ao usuário (como erros na carga do programa).

3. Conclusão

O simulador do processador MIPS I, e outros simuladores gerados pelo *framework* (como as versões monociclo e multiciclo do MIPS I, e o processador Neander [Weber 2001]) vêm sendo empregados no ensino de Organização e Arquitetura de Computadores na Universidade de Caxias do Sul desde 2003, obtendo pareceres favoráveis por parte dos estudantes. Eles também são empregados em pesquisa e, atualmente, um novo recurso, que envolve compressão de código para a redução do consumo de energia em processadores embarcados, está sendo investigado. Mais detalhes sobre a aplicação do *framework* podem ser encontrados em [Soares and Wagner 2006].

A integração do simulador do processador MIPS I com o compilador GNU gcc, que já está sendo aplicada em pesquisa, será usada no ensino de Organização e Arquitetura de Computadores no próximo período letivo. Este uso no ensino prevê, igualmente, disciplinas de Programação de Computadores, para ilustrar a técnica de compilação.

O *framework* de modelagem está disponível no endereço *http://hermes.ucs.br/carvi/cent/dpei/snsoares/TDBench*. Ele foi desenvolvido com a linguagem Java e, por isso, é independente de plataforma, além de ser uma ferramenta de código aberto.

4. Referências

Laurus, J. (2004) "SPIM: A MIPS R2000/R3000", http://www.cs.wisc.edu/~larus/spim.html, March.

Uy, R. L., Bernardo, M. and Josiel, E. (2004) "DARC2: 2nd. Generation DLX Architecture Simulator", Proceedings of the Workshop on Computer Architecture Education, Munich, Germany, p. 99-104.

Verplaetse, P., Campenhout, J.V. and Neers, H. (1999) "ESCAPE: Environment for the Simulation of Computer Architectures for the Purpose of Education", IEEE TCCA Newsletter, Los Alamitos, p. 57-59.

Soares, S. and Wagner, F.R. "Design Space Exploration of Embedded Processors in Computer Architecture Education using T&D-Bench", Proceedings of the 36th ASEE/IEEE Frontiers in Education Conference (a ser publicado).

Weber, R. F. (2001) "Fundamentos de Arquitetura de Computadores", In: 2.ed. Porto Alegre: Instituto de Informática da UFRGS: Sagra Luzzatto, (Série Livros Didáticos, n. 8), 2001. 299 p.