Projeto A*

Douglas Juares Ortlieb, Reinaldo Teixeira
Coordenação do Curso de Bacharelado em Ciência da Computação - COCIC
Universidade Tecnológica Federal do Paraná - UTFPR
Campus Campo Mourão
Campo Mourão, Paraná, Brasil
ortliebd@alunos.utfpr.edu.br e segundo@email.com.br

Resumo

(Esboço)Este artigo tem como objetivo apresentar o processo de transformação de um código de alto nível para código de máquina, transformando um programa em C para assembly (MIPS) e em seguida para linguagem de máquina e também na implementação de uma ferramente capaz de converter código de máquina em assembly para MIPS.

1 Introdução

Linguagens de programação de alto nível foram criadas com a intenção de facilitar a escrita de programas para computadores, buscando uma aproximação com as linguagens naturais, possibilitando ao programador manter o foco principalmente na resolução do problema.

Contudo, as linguagens de alto nível não podem ser executadas diretamente no computador e precisam de uma codificação na qual o computador as entenda. Essa codificação é conhecida como sistema de numeração binária, formada por zero e um (0 e 1) ou nível de tensão alto e baixo.

Neste artigo explicaremos o processo de transformação utilizando um programa escrito na linguagem C, alto nível, convertendo-o para *assembly* e em seguida para linguagem de máquina, onde finalmente poderá ser executado.

As linguagens *assembly* representam um nível intermediário de instruções, utilizam um montador para traduzir uma notação simbólica para binário e são escritas para arquiteturas específicas de microprocessadores, portanto faremos uso de um processador MIPS (*Microprocessor without Interlocked Pipeline Stages* - Microprocessador com Estágios de Pipeline Integrado) do tipo RISC (*Reduced Instruction Set Computer* - Computador com Conjunto de Instruções Reduzidos) dado a sua simplicidade e seus fins didáticos.

Também é possível realizar o processo reverso, ou seja, transformar um código binário em *assembly* e de *assembly* para uma linguagem de alto nível. Parte deste trabalho consiste na implementação de um *disassembler* (desmontador) em C responsável por converter o código binário em *assembly*.

2 MIPS

Ao contrário dos programas nas linguagens de alto nível, os operandos das instruções aritméticas são restritos, precisam ser de um grupo limitado de locais especiais, embutidos diretamente no hardware, chamados *registradores*. Uma diferença importante entre as variáveis de uma linguagem de programação e os registradores é o número limitado de registradores, o MIPS possui 32 registradores com tamanho de 32 bits cada[1].

Tabela 1: Registradores do MIPS

Nome	Número	Uso
$zero	0	O valor constante 0
$at	1	Temporário do montador
$v0-$v1	2-3	Valores para resultados de função e avaliação de expressão
$a0-$a3	4-7	Argumentos
$t0-$t7	8-15	Temporários
$s0-$s7	16-23	Temporários salvos
$t8-$t9	24-25	Temporários
$k0-$k1	26-27	Reservados para kernel do SO
$gp	28	Ponteiro global
$sp	29	Stack Pointer
$fp	30	Frame Pointer
$ra	31	Endereço de retorno

*Trabalho desenvolvido para a disciplina de BCC33B – Arquitetura e Organização de Computadores.

Existem três tipos de instruções no MIPS. As instruções do tipo Registrador Tipo-R (*R-Type*) realizam operações aritméticas e lógicas com dados buscados do banco de registradores, salvando no mesmo a resposta da operação. Esse tipo de instrução possui 6 campos, o código do campo *Opcode*, de 6 bits, junto com o campo *Funct*, também de 6 bits, informam qual operação será realizada. Os campos *Rs* e *Rt* indicam os registradores de origem e o campo *Rd* indica o registrador de destino da operação, o campo *Shamt* informa quantos bits devem ser deslocados, para esquerda ou para a direita, todos possuem 5 bits. Instruções do Tipo-I (*I-Type*) são utilizadas de várias formas - operações aritméticas e lógicas, desvios condicionais, leitura e escrita na memória de dados -

Referências

1 PATTERSON, David A.; HENNESSY, John L. *Organização e Projeto de Computadores: a interface hardware/software*. Rio de Janeiro: Elsevier, 2005.