Estudo da redundância na execução do *benchmark* MiBench para a arquitetura ARM*

Giovane de O. Torres†, Rodrigo C. de Moura, Maurício Lima Pilla

Centro de Desenvolvimento Tecnológico
Universidade Federal de Pelotas (UFPEL) – Pelotas, RS – Brasil

{gdotorres, rcmoura, pilla}@inf.ufpel.edu.br

Resumo. *Este artigo propõe avaliar a redundância existente na execução do* benchmark *MiBench sobre a arquitetura ARM, sendo uma continuação de outro trabalho relacionado à análise do perfil de consumo do MiBench. O simulador Sim-Panalyzer será usado para gerar traços com endereço, entradas e saídas com a intenção de buscar por repetições dos traços para avaliar o comportamento da técnica de reuso de traços.*

1. Introdução

Atualmente, existe uma grande difusão de dispositivos portáteis, como por exemplo *smartphones*, PDAs, *tablets* e calculadoras. Nestes, é possível encontrar uma categoria de arquitetura embarcada do tipo ARM (*Advanced RISC Machine*), que necessita atender um requisito importante, o de possuir um consumo de energia reduzido com bom desempenho.

O artigo atual é uma continuação de um trabalho que se preocupou em verificar os perfis de consumo de energia das aplicações do MiBench, relacionando o total de energia dissipada e o número de instruções executadas por ciclo [Torres et al. 2012]. O principal objetivo deste trabalho é de procurar avaliar a redundância que existe na execução do MiBench. O simulador Sim-Panalyzer será usado para gerar traços, nos quais serão procurados repetições. Com isto, será possível avaliar como se comportará o mecanismo de reuso de traços.

Esse artigo divide-se nas seguintes Seções. Na Seção 2, a arquitetura ARM é apresentada. Com a finalidade de avaliar o efeito do reuso sobre uma arquitetura ARM, serão gerados traços de instruções, cuja técnica de reuso será explicada na Seção 3, bem como entradas e saídas usando o simulador Sim-Panalyzer. Este simulador será contextualizado na Seção 4. A Seção 5 deste artigo será dedicada a discussões sobre o trabalho em si, onde será mostrado como o trabalho será realizado, além de considerações sobre o principal objetivo do artigo corrente.

2. Arquiteturas ARM

2.1. Características gerais

Esta proposta de trabalho utiliza-se como base as arquiteturas do tipo ARM, as quais são do gênero superescalar [Fernandes e Santos 1997]. Na prática, isso significa que a arquitetura ARM possui o paralelismo implementado no *hardware*. É importante ressaltar que,

*Projeto PRONEX/FAPERGS/CNPq GREEN-GRID Computação de Alto Desempenho Sustentável
†Bolsista FAPERGS PIBIC

esta arquitetura é considerada largamente popular [Ryzhyk 2006], devido aos seguintes aspectos:

- Se forem comparados um *core* do tipo ARM com outro de utilização geral, o primeiro é mais simples. Com isso, é possível construir um processador ARM com um número relativamente pequeno de transistores, permitindo a inserção de uma quantidade maior de componentes voltados para aplicações específicas;
- A arquitetura ARM tem o foco na questão de consumir a menor quantidade de energia possível;
- Esta categoria de arquitetura é altamente modular. Isto indica que todos os componentes que constituem um processador ARM, com exceção do *pipeline* de inteiros, são opcionais no que diz respeito a utilização dos mesmos.

2.2. Especificações Técnicas

Esta espécie de arquitetura é do tipo RISC (*Reduced Instruction Set Computer*), e possui um total de 37 registradores [Seal 2000]. Destes, 31 são direcionados para utilização geral, enquanto os 6 restantes são registradores de *status*.

Além disso, é importante verificar os tipos de instruções de uma arquitetura ARM. Pode-se categorizar as instruções de um processador ARM em 6 grandes classes [Gomes et al. 2005], as quais são as seguintes:

- Instruções de processamento de dados;
- Instruções de *Load* e *Store*;
- Instruções do tipo *branch*;
- Instruções de transferência de registradores de *status*;
- Instruções de co-processador;
- Instruções geradoras de exceções.

As instruções possuem tamanho fixo, com 32 *bits*, mas a arquitetura ARM ainda é capaz de executar instruções de 16 *bits*, utilizando-se do conjunto de instruções Thumb, as quais tornam o código mais compacto.

Uma particularidade importante relacionada às instruções ARM refere-se aos bits de condições. A maioria das instruções ARM possui 4 bits, que podem determinar a execução ou não de uma instrução. Estes quatro bits possibilitam a existência de 16 situações condicionais. Porém, é importante ressaltar que uma destas situações condicionais é a que a instrução sempre executará. As outras instruções podem ser utilizadas, por exemplo, para condições tais como igualdade, desigualdade, maior que e menor que.

3. Reuso de Traços

A técnica que é o foco principal deste artigo, consiste no reuso de sequências dinâmicas de instruções (traços), os quais possuem um contexto de entrada e saída [Laurino 2007]. Este mecanismo possui o objetivo de evitar a re-execução individual das instruções que estão dentro de um traço [González et al. 1999]. Este método pode ser melhor contextualizado se for explicado primeiramente outra técnica, a de reuso de blocos básicos.

Um bloco básico consiste de uma sequência dinâmica de instruções, e se existir alguma instrução de desvio (*Branch*), esta será a última instrução do bloco. Este

mecanismo utiliza-se de uma tabela, a qual é chamada de *Block History Buffer* (BHB) [Huang e Lilja 2000]. Nesta tabela são gravados valores de entrada e saída do bloco, bem como o endereço para o próximo bloco básico. Além disso, se existirem instruções de acesso e/ou leitura na memória, a BHB também se encarrega de manter esses tipos de instruções registrados. O método de reuso de blocos básicos pode ser definido mostrando os seguintes passos:

1. Se um bloco é obtido, a BHB é consultada para procurar as instâncias do mesmo. Caso não encontre nada, nenhuma ação é feita;
2. Ao encontrar alguma instância do bloco, os valores que estão armazenados nos registradores de entrada são comparados com os que estão gravados na BHB;
3. Na ocorrência de instruções as quais utilizam-se de leitura na memória, é necessário uma consulta à memória *cache*, e os dados armazenados na BHB também são comparados com os da *cache*;
4. Após feita a comparação entre os valores dos registradores de entrada, mais os da cache e os gravados na BHB, conclui-se que é possível reusar a instância do bloco básico. Se não for possível reusar o bloco, então a instrução a seguir é despachada normalmente.

O reuso de traços possui uma abordagem semelhante ao reuso de blocos básicos, porém existe uma importante diferença entre eles. Os blocos básicos são limitados a terem somente uma única instrução de desvio, e esta deve ser a última do bloco. Já em traços, essa restrição não existe, podendo conter mais de uma instrução do tipo *branch*. Um dos mecanismos mais conhecidos de implementação do reuso de traços é o *Reuse through Speculation on Traces* (RST) [Pilla 2004], que envolve não somente a reutilização de traços, mas também inclui método especulativo para formação de traços.

4. Sim-Panalyzer

O simulador Sim-Panalyzer [Mudge et al. 2001] é o *software* empregado para simular uma arquitetura ARM em um computador de uso geral. Esta ferramenta foi baseada em um outro *software*, chamado de SimpleScalar [Austin 1997], que é voltado para simular arquiteturas tais como do tipo x86, PISA, Alpha, além de ARM.

O principal objetivo da ferramenta Sim-Panalyzer é a de criar um estimador adiantado de potência, permitindo avaliar ganhos e perdas na relação entre desempenho e dissipação de energia, o que permitirá efetuar uma análise detalhada de como a técnica de reuso de traços afeta o desempenho de uma arquitetura ARM.

5. Discussões

O presente artigo tem como proposta estudar como o método de reuso de traços pode influenciar no desempenho de uma arquitetura ARM.

Para esta análise ser feita, primeiramente será feito um registro das entradas e saídas dos blocos processados das aplicações as quais serão executadas sobre uma arquitetura ARM. Logo, esta divisão do trabalho se dedicará a simular *benchmarks* sobre o *software Sim-Panalyzer*.

Em um segundo momento, é necessário realizar uma verificação dos blocos, com o intuito de analisar quantas vezes esses serão executados com as entradas e saídas repetidas, efetuando deste modo um pós-processamento dos blocos analisados. Em um último

instante, a estratégia de reuso de traços será testada em uma arquitetura ARM, o que possibilitará fazer uma avaliação prática deste mecanismo na arquitetura.

O artigo encontra-se na fase atual de finalização do estudo dos materiais que serão utilizados no trabalho, para após começar a simulação dos *benchmarks* do MiBench.

Referências

Austin, T. (1997). *SimpleScalar LLC*. Disponível em: <http://www.simplescalar.com/>. Acesso em: dezembro de 2012.

Fernandes, E. S. T. e Santos, A. D. (1997). *Arquiteturas super-escalares: detecção e exploração do paralelismo de baixo nível*. Instituto de Informática da UFRGS.

Gomes, P. H., Leite, T. S., e Caetano, U. I. (2005). *A Arquitetura ARM*. Disponível em: <http://www.ic.unicamp.br/∼rodolfo/Cursos/mc722/2s2005/Trabalho/g20-arm.pdf>. Acesso em: dezembro de 2012.

González, A., Tubella, J., e Molina, C. (1999). *Trace-level Reuse*. Proceedings of the the International Conference on Parallel Processing.

Huang, J. e Lilja, D. J. (2000). *Exploring Sub-Block Value Reuse for Superscalar Processors*. Proceedings of 2000 International Conference on Parallel Architectures and Compilation Techniques (Pact'00) October 15 - 19, 2001.

Laurino, L. S. (2007). *Reuso Especulativo de Traços com Instruções de Acesso à Memória*. 99f. Tese (Mestrado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

Mudge, T., Austin, T., e Grunwald, D. (2001). *The SimpleScalar-Arm Power Modeling Project*. Disponível em <http://web.eecs.umich.edu/∼panalyzer/>. Acesso em: dezembro de 2012.

Pilla, M. L. (2004). *Reuse through Speculation on Traces*. 174f. Tese (Doutorado em Ciência da Computação) - Programa em Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre.

Ryzhyk, L. (2006). *The ARM Architeture*. Disponível em: http://www.cse.unsw.edu.au/∼cs9244/06/seminars/08-leonidr-t.pdf>. Acesso em: dezembro de 2012.

Seal, D. (2000). *Arm Architecture Reference Manual*. Addison-Wesley Longman Publishing Co.

Torres, G. O., Natchigall, M. G., e Pilla, M. L. (2012). *Consumo de Aplicações do MiBench*. XXI Congresso de Iniciação Científica - UFPEL, Novembro de 2012.