Parallelizing Load/Stores on Dual-Bank Memory Embedded Processors

Revista TEC – Volume, issue 6 (agosto 2006) - Pages: 613-657
Autores: XIAOTONG ZHUANG e SANTOSH PANDE
Aluna: Andréia Rodrigues Casare RA 050367

Resumo:

Muitos processadores embutidos modernos com apoio de DSPs dividiram bancos de memória, chamada de memória de dual-banco, junto com load/stores paralelas para melhorar o desempenho. Para utilizar as ordens load/stores paralelas efetivamente, o compilador tem que dividir os valores de memória residente e os nomear para banco X e banco Y.

Exemplos de processadores que suportam dual-banco, Motorola DSP56000 série e Sony pDSP.

O que eles tentam é a fusão de load/stores sem duplicação e divisão de teias.

Para separar referências de memórias, eles usam o conceito de "Teia". Cada teia constrói um símbolo de separação. Dessa maneira variáveis de separação em teias, alcançam separação de valor efetiva e melhoram a tarefa do banco. Separação de valor nos oferece mais liberdade nomeando bancos e oportunidade de load/stores paralelas crescentes.

Quando construímos teias, determinamos a gama de movimento para todas as ordens de load/stores e construímos o MSG, que reflete ordens potenciais que podem ser fundidas.

O MSG tem os seguintes conflitos: Conflitos de endereço e conflito de registro. Para resolver esses conflitos, são usados dois algoritmos heurísticos. O primeiro é o Algoritmo Simulated Annealing (AS), nesse algoritmo o tempo mais longo é a melhor solução e o segundo algoritmo, é um algoritmo ganancioso que procura o espaço de solução acrescentando grupo de nodo gradualmente a dois determinados jogos (DETX para o banco X e DETY para o banco Y).

São citadas várias maneiras de melhorar o desempenho, como duplicações e otimizações do compilador para aumentar a efetividade de aproximação.

A duplicação de instrução (Cross-BB Merge) duplica ordens de load/store por blocos básicos, isso pode criar novas oportunidades de combinações. Cross-BB pode unir load/stores em blocos básicos diferentes se eles estão em bancos diferentes.

A Duplicação Variável é feita com a intenção de armazenar algumas variáveis em ambos bancos de memória, podem ser combinadas instruções de load destas variáveis com outras load/stores de outros bancos. Eles notaram que a duplicação sempre não é lucrativa, porque devem ser acrescentadas ordens de store adicionais a toda store na teia.

Web splitting corta as teias em tamanho menor e criar mais chances pelas ordens de load/stores na teia a ser fundida com outras ordens de acesso de memória. Como load/stores na mesma teia deve ser alocado no mesmo local de memória, significa que eles devem ser colocados no mesmo banco de memória. Teias grandes podem proibir load/stores de se fundir por causa da restrição do banco de memória.

Para variáveis globais, podem ser determinados só uma vez nos bancos. Depois que o banco for decidido em um procedimento, outros procedimentos têm que tratar isto como uma variável fixa do banco.

Na otimização global por procedimentos de clonagem, a passagem de parâmetros é diferente para vários visitantes, são criadas cópias diferentes para cada visitante. Isto permite mais oportunidades para combinar load/stores para cópias de procedimentos clonados. Esse tipo de otimizaçãoglobal pode causar aumento de tamanho de código dramático, só devem ser executadas tais otimizações para funções pequenas ou funções invocadas raramente.

Otimizações perfil-acumuladas nos dão mais conhecimento da probabilidade que cada caminho levará e cada instrução que será executada.

Conclui-se que aumentando a gama de movimento de ordens sistematicamente e duplicação de ordens e variáveis junto com divisão de teias, pode gerar qualidade de código comparável.

Os passos de otimização podem ser guiados perfilando informação para impulsionar desempenho do tempo corrido significativamente enquanto mantendo os dados e crescimento de código e tempo de compilação quase não afetado.