Achieving Out-of-Order Performance with Almost In-Order Complexity

Francis Tseng Yale N. Patt
Department of Electrical and Computer Engineering
The University of Texas at Austin
{ tsengf,patt} @hps.utexas.edu

International Symposium on Computer Architecture 2008

Aluna: Gabriela Batista Leão - RA:087348 – gabileao@gmail.com

Resumo

A quantidade de instruções despachada fora de ordem nas arquiteturas modernas pode ser ainda bastante aumentada, utilizando-se métodos de otimização tradicionais, a fim de explorar melhor o desempenho. No entanto, para refletir essas melhorias, adaptações devem ser implementadas diretamente no hardware, geralmente no custo do aumento da complexidade de seu projeto e requerimentos de potência. Como estes últimos fatores são fortes limitadores de desempenho, o trabalho apresentado propõe uma abordagem combinada de implementação no nível do compilador e da microarquitetura, considerando a máquina Alpha, cuja arquitetura contém o conjutno de instruções ISA. A abordagem proposta define, inicialmente, a granularidade das instruções (tarefa), estas representam um determinado código-fonte de uma linguagem de alto nível, a ser compilada, considerando análise de desempenho das aplicações na suíte SPEC CPU2000, que foi compilada para a arquitetura já referida. Esta análise se concentra no comportamento dos valores das variáveis ao longo da execução das aplicações, nomeadamente ao *fanout,* definido pela quantidade de vezes que um valor é lido, e ao seu tempo de vida, medido pela quantidade de instruções executadas entre sua produção e consumo. Os principais resultados dessa análise mostram, em média, que 90% dos valores é lido no máximo 2 vezes (*fanout*) e 80% dos valores tem tempo de vida de 32 instruções ou menos. Notando-se que tanto o *fanout* quanto o tempo de vida são relativamente pequenos, o grafo do fluxo de dados do programa pode ser dividido em subgrafos distintos. Cada subgrafo contém uma unidade, cuja granularidade é o *braid.* Cada *braid* abrange uma operação desempenhada em linguagem de alto nível, i.e. o *braid* reside somente dentro de um bloco básico. Essa abordagem requer (1) que o compilador seja capaz de construir *braids* a partir do grafo do fluxo de dados do programa. Além disso, (2) a ISA precisa transmitir informações do compilador para a microarquitetura e, finalmente, (3) a microarquitetura deve ser consciente de *braids* para tirar proveito destes últimos. Como *braid* é uma entidade identificada em tempo de execução, para cumprir o requerimento (1) o compilador deve analisar o grafo do caminho de dados do programa e armazenar o produtor e o consumidor de cada valor produzido. É também durante este passo que o compilador sabe que valores serão usados dentro e fora do bloco básico. A partir desses perfis, o compilador deve identificar os diferentes *braids* usando um algoritmo de coloração de grafos simples. A formação de um *braid* é realizada pela seleção de uma instrução dentro de um bloco básico e a identificação do subgrafo correspondente a essa instrução. Esse procedimento é repetido até que a última instrução do bloco básico seja selecionada e adicionada ao *braid* respectivo. Na sequencia, os *braids* são rearranjados de tal forma que instruções pertencentes a um mesmo *braid* sejam executadas sequencialmente. Finalmente, registradores externos são alocados primeiro e só depois a alocação individual de registradores de cada *braid* é realizada. Para transportar informações dos *braids* anteriormente definidos, (2), foram acrescentadas 3 instruções extendendo a ISA, que contêm como campos o bit de início do *braid,* um bit associado com cada operando de origem que especifica se o valor usado é carregado de um registrador interno ou externo ao *braid* e um bit associado ao operando de destino que definde se o resultado da instrução deve ser escrito em um arquivo de registrador interno ou externo. Por último, em (3) a microarquitetura deve refletir a alocação de registradores em 2 passos como já referenciado. A principal mudança na microarquitetura é a implementação de Unidades de Execução de Braids (BEUs), responsáveis por interpretarem as instruções descritas em (2). Conclui-se que a proposta melhora o desempenho em 9%, com complexidade de projeto similar a de instruçãoes em ordem, por simplificar o projeto de hardware nas fases de alocação, já que resultados de instruçãoes que escrevem para registradores internos não precisam ser escritos em registradores externos; renomeação, pois operandos internos não precisam ser renomeados; e um arquivo de registrores externos mais simples. A utilização das BEUs também remove parte da complexidade da rede de sobreposição.