Aluna: Carla Geovana do Nascimento Macário **RA:**895063

RENO: A Rename-Based Instruction Optimizer In *ISCA '05: Proceedings of the 32nd Annual International Symposium on Computer Architecture*, pags. 98 – 109. IEEE Computer Society Washington, DC, USA.2005

RENO (REnaming Optimzer) é um mecanismo de renomeamento de registros do MIPS-R10000 modificado para implementar dinamicamente algumas técnicas de otimização estática existentes, promovendo a eliminação de instruções e levando à diminuição na latência do grafo de fluxo de dados e ao aumento do *bandwidth* do core de execução. Com a implementação dinâmica, em contraposição à estática, tem-se maior espaço físico para renomeamento de registradores, inexistência de limitações presentes nos compiladores, otimização baseada em informação especulativa ou dinamicamente disponível sobre dependência de memória e executada apenas nos caminhos dinâmicos. Em caso de mis-especulação, ou de informação de memória errada, as instruções executadas são desfeitas, bem como as otimizaçoes executadas pelo RENO.

RENO unifica um novo mecanismo ($RENO_{CF}$) com outros previamente propostos ($RENO_{ME}$, $RENO_{CSE}$ e $RENO_{RA}$), promovendo uma sinergia entre eles. $RENO_{ME}$ elimina *moves* e é a mais simples das otimizações disponíveis. Requer, além de uma infra-estrutura de compartilhamento de registradores, um circuito para identificar *moves*. No renomeamento, o registrador de entrada de uma instrução *move* passa a apontar para o registrador correspondente à sua saída, eliminando a instrução, que deixa de ser executada. O $RENO_{CSE}$ faz a eliminação de subexpressões comuns, mantendo uma tabela com os valores disponíveis em cada registrador físico e com informaçoes do fluxo de dados da instrução que levou a este valor. No renomeamento é feita uma busca por tuplas com o mesmo código de operação que a instrução corrente. A existência desta tupla indica redundância e a instrução deve ser eliminada, passando sua saída a apontar para a saída do registrador correspondente à tupla identificada. O $RENO_{RA}$ implementa dinamicamente a alocação de registros, tendo como principal objetivo eliminar *loads* via integração de registradores (com o compartilhamento de registradores físicos). É usado para implementar especulação de *bypassing* de memória para pares de *load-store,* transformando cadeias produtor-store-load-consumidor em produtor-consumidor. O $RENO_{CF}$ implementa dinamicamente a técnica de *folding* de constantes em adições do tipo *register-immediate*, escolhidas por serem mais comuns nos programas (cálculo de endereços, etc.) e por gerarem operações menos complexas após o *folding* (com 2 operandos). Usa uma extensão do formato da tabela de mapeamento, armazenando, além das informações convencionais, um deslocamento que contém o valor resultante das adições. No mapeamento um acumulador armazena a(s) soma(s) do valor do registrador com o deslocamento, disponibilizando o resultado assim ele que for requisitado por alguma instrução subseqüente. Desta forma, e de maneira semelhante às técnicas anteriores, instruções intermediárias são eliminadas.

A implementação do RENO é simples, usando sempre o mesmo princípio: manipulação de tabelas de mapeamento e infra-estrutura de compartilhamento de registros físicos para promover o colapso (ou eliminação) de instruções dinâmicas (diminuindo a latência do fluxo de dados). A instrução eliminada é colocada num buffer de reordenamento, do qual poderá ser sertirada mas não executada (diminuindo o core de execução). Em relação ao MIPS RS1000, RENO usa um renomeador convencional e adiciona uma lógica de seleção de saída para cada slot de instrução. Há ainda um circuito paralelelo à estrutura do renomeador que acumula e seleciona deslocamentos no lugar de nomes de registros físicos. Na avaliação do RENO foram utilizados os programas de benchmark SPECint e MediaBech, executando numa arquitetura Alpha EV6 com o compilador OSF Digital com otimizações -O4. O $RENO_{ME}$ eliminou em média 4% das instruções dinâmicas e o $RENO_{CF}$ eliminou adicionalmente 12%(SPECint) e 16%(MediaBench). Considerando também o $RENO_{CSE/RA}$, o ganho a mais foi de 5%(SPECint) e 3%(MediaBench), o que em se tratando de *load*, é um importante. No geral as eliminações promoveram ganhos de performance de 8%(SPECint) e de 13%(MediaBench), variando conforme as instruções predominantes nos respectivos caminhos críticos (*load* ou operações de ALU).