Título: Exploiting Vector Parallelism in Software Pipelined Loops
Autores: Samuel Larsen, Rodric Rabbah e Saman Amarasinghe
MIT Computer Science and Artificial Intelligence Laboratory
Congresso: Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture, páginas 119-129, 2005.

Resumo por Renato Silva das Neves – RA 057639

Os processadores atuais têm adicionado ao seu ISA instruções que operam sobre vetores, principalmente no caso de instruções multimídia. Uma maneira de otimizar o uso dos processadores, aumentando sua performance, é justamente explorar as características de paralelismo vetorial entre *loops* no código com *pipeline* de *software* (*software pipelining*). Essa é a proposta do artigo analisado, que utiliza a vetorização seletiva (*selective vetorization*) para dividir operações entre recursos escalares e vetoriais do processador, de uma maneira que maximize a performance quando um *loop* está em *pipeline* de *software*. Assim, em *loops* com um grande número de operações vetoriais, estratégias convencionais fazem com que todas operações de dados paralelas sejam vetoriais, deixando recursos escalares ociosos. Movendo algumas operações para as unidades escalares pode se fazer com que o *scheduling* seja mais efetivo.

O algoritmo utilizado para a vetorização seletiva foi baseado na heurística de particionamento em dois *clusters* de Kernighan e Lin's, dividindo instruções entre partições escalares ou vetoriais. O algoritmo é iterativo e funciona da seguinte forma: inicialmente todas operações são colocadas na partição escalar e cada iteração reposiciona cada operação que pode ser vetorial exatamente uma vez. Com cada movimento, o algoritmo calcula o custo do resultado da reconfiguração, registrando o custo mínimo encontrado. Uma vez que cada operação foi reparticionada, a configuração com o menor custo é usada como o ponto de partida da próxima iteração. O processo termina quando uma iteração falha ao tentar melhorar sua configuração inicial. No final, operações que ficaram na partição vetorial são passíveis de vetorização. O cálculo do custo da configuração é a principal parte do algoritmo e é definida como o peso do recurso mais usado, ou seja, o número de ciclos de máquina que o recurso mais usado está reservado. O cálculo desse custo é feito a partir do método *bin-packing*, onde cada operação adicionada em um *bin* (recurso visível ao compilador) aumenta o peso desse *bin*. Uma otimização efetuada na proposta do artigo foi que quando dois *schedulings* alternativos não aumentam o peso do recurso mais usado, a opção escolhida é a que minimiza a soma dos quadrados dos pesos dos *bins*. Isso gera um balanço de operações entre os *bins*. No pior caso, o algoritmo de vetorização seletiva resulta em uma complexidade de $O(n^3)$.

O algoritmo acima foi implementado no *beckend Trimaran*, uma infra-estrutura de compilação e simulação para arquiteturas VLIW. Também foi usado o *frontend* SUIF. Para todos os *benchmarks* utilizados (nove SPEC FP) foram aplicados um conjunto de otimizações padrões antes da vetorização seletiva, que só foi aplicada em *loops do*. A comparação de *speedup* foi realizada com um método de vetorização tradicional e outro de vetorização completa (*full*). A vetorização seletiva alcançou o máximo de 1.38x de *speedup*, com uma média de 1.11x, mostrando melhor performance em todos os casos, exceto em um, em que foi criado mais *schedules* compactos para *loops* críticos, aumentando o número de estágios do *pipeline* de *software*, levando a prólogos e epílogos mais longos. Resultados mostraram que para cada *benchmark* há um significante número de *loops* no qual a vetorização seletiva fornece vantagens. A comunicação de operandos entre recursos escalares e vetoriais, a partir de instruções *load* e *store*, é levada em conta nos cálculos do custo descritos acima e resultados também foram mostrados comprovando que considerando tal comunicação há uma melhora no *speedup*. Por último, as operações vetoriais de memória podem ser consideras alinhadas ou não alinhadas. Simulações mostraram que o *overhead* de alinhamento pode ser eliminado quando as operações são conhecidas em tempo de compilação como sendo alinhadas, pois assim não são consideradas durante a análise de custo do algoritmo de vetorização seletiva.