Análise da influência da reordenação de instruções no desempenho de processadores

Felipe Salerno Prado

Universidade Federal do Rio Grande do Sul
Instituto de Informática
Orientado por Prof. Dr. Antonio Carlos S. Beck Filho
fsprado@inf.ufrgs.br

Motivação

Atualmente os **processadores superescalares** utilizam uma quantidade desproporcional de área (mensurada em número de transistores) e potência, quando comparada ao retorno em termos de desempenho que este aumento oferece. Por isso, mesmo com processadores superescalares, grande parte dos processadores presentes em **sistemas embarcados** continuam adotando a **execução de instruções em ordem**, devido a menor complexidade do hardware e menor consumo de energia. Álem disso, os recursos e características que diversos processadores superescalares têm são diferentes entre si (e.g.: a quantidade disponível de unidades funcionais, o tempo para processar cada operação, etc.), entretanto o compilador e o programa gerado são geralmente os mesmos.

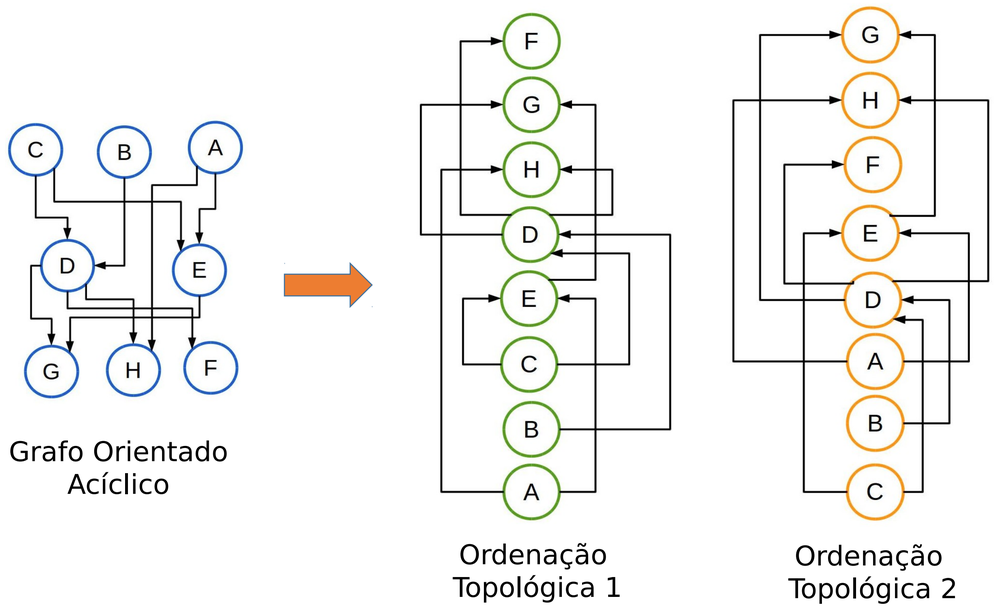
- **Conseguir aumentar o desempenho, mesmo que seja em pequenas proporções, sem inserir custo adicional ao hardware ou consumo de energia.**

Solução Proposta

Explorar o efeito do **reordenamento de instruções** em programas, mantendo a lógica, mas com blocos básicos cuja ordem de suas instruções diferem das geradas pelo compilador. O objetivo é analisar a mudança de desempenho ao realizar pequenas modificações no programa compilado em Assembly, antes de gerar o executável.

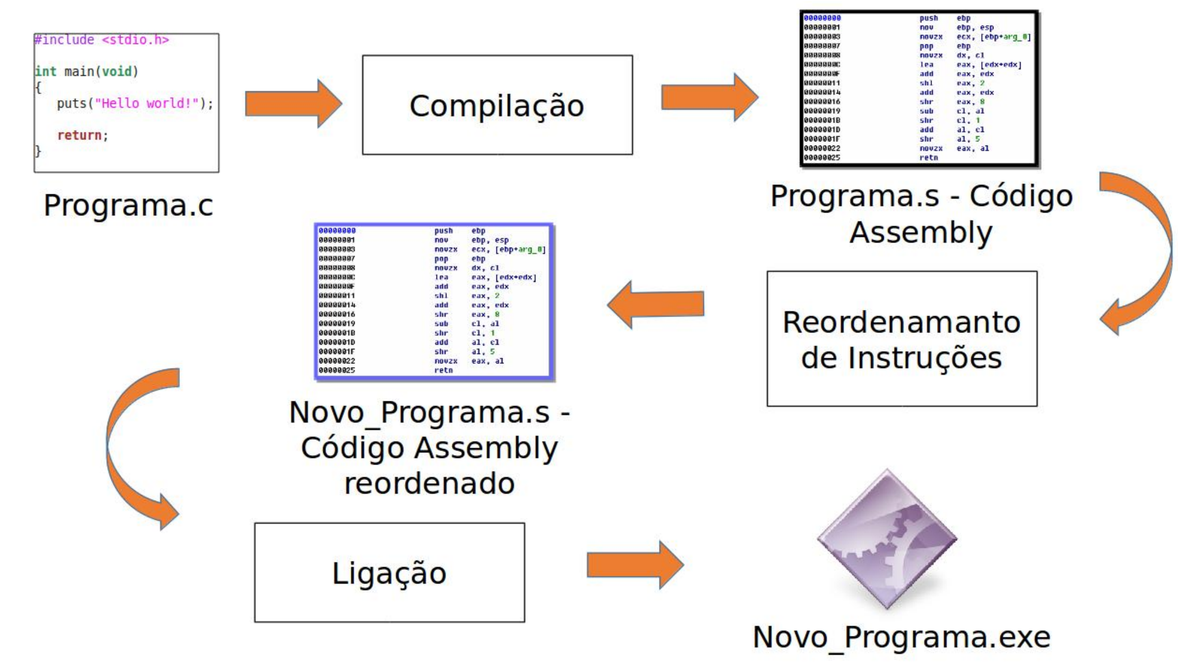
Metodologia

Podemos considerar que um programa pode ser dividido em diversos blocos básicos (i.e.: uma sequência de instruções sem operações de desvio). Dependências entre instruções existem dentro de um bloco básico, e, para a lógica do programa estar correta, é necessário que elas sejam respeitadas. Assim, a reordenação de instruções proposta tem de ser feita respeitando os limites de cada bloco básico. Através da análise de dependências entre instruções dentro de um bloco básico é possível modelar e linealizar (ordenação toplógica) um grafo, representando cada nodo como uma instrução e cada aresta como uma dependência entre duas instruções. Por meio desta avaliação, pode-se obter uma ou mais **ordenações topológicas** de instruções de um mesmo bloco básico, o que resulta em diferentes programas com lógica idêntica.



Exemplo de Ordenação Topológica.

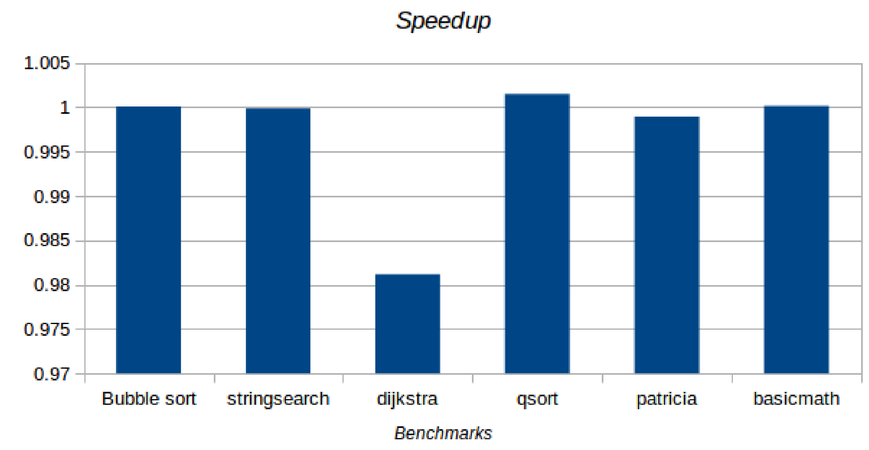
O *parser* das instruções e a obtenção de diferentes ordenações topológicas dos blocos básicos dos arquivos foram desenvolvidas na linguagem de programação Java. Esses programas foram executados no simulador Multi2Sim, adaptado para execução de instruções em ordem.



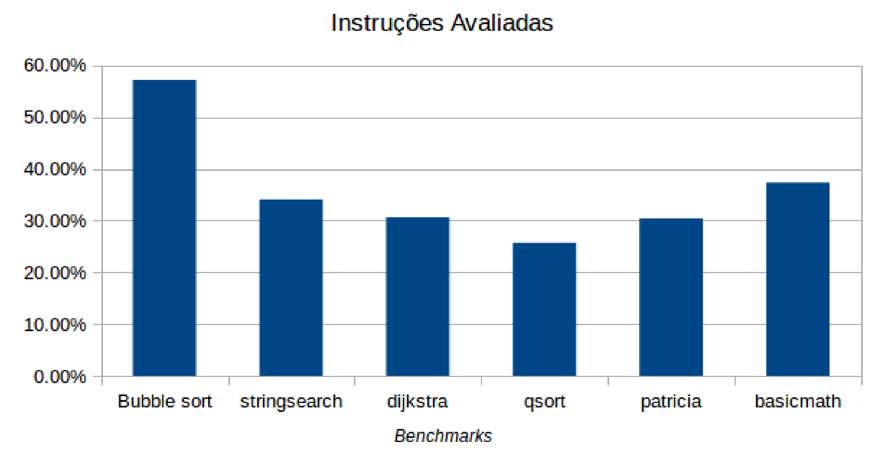
Processo de geração de programas com instruções reordenadas.

Resultados

O método foi aplicado a alguns programas do MiBench, um conjunto de *benchmarks* para sistemas embarcados. A reordenação proposta foi feita de forma **randômica**, com 20 versões para cada programa escolhido. Os resultados mostram que os programas com instruções reordenadas apresentam um desempenho muito semelhante ao do original, o que mostra que a randomização não modificou o cenário do código-fonte gerado pelo compilador, que não leva em consideração os recursos do processador.



Speedup da média dos programas modificadas em relação ao respectivo programa original.



Porcentagem de instruções avaliadas em cada programa.

Conclusões e Trabalhos Futuros

Através das diferentes aplicações e do número de simulações realizadas é possível concluir que a solução proposta se provou ser pouco relevante quanto a mudança de desempenho. Para trabalhos futuros, seria possível observar a relação entre as intruções de cada bloco básico com a disponibilidade dos recursos do processador superescalar e, portanto, reordenar as instruções conforme essa relação. Dessa maneira, uma execução em ordem seria realizada com uso mais eficiente dos recursos do processador.

SIC UFRGS 2015. XXVII Salão de Iniciação Científica - UFRGS, 19 a 23 de outubro de 2015. Campus do Vale, Porto Alegre, Brasil.