Simulação de PowerPC utilizando MPSoCBench

Kelly Sandim, Lucas Queiroz, Vitória Freitas

[1]Faculdade de Computação (FACOM)
Universidade Federal de Mato Grosso do Sul (UFMS)
Campo Grande – MS – Brazil

29 de março de 2022

1. Introdução

O ato de simular um programa em um processador expõe suas características e desempenho, mostrando dados práticos da execução de aplicações amplamente conhecidas. A simulação possibilita obter, de forma rápida e segura, informações em tempo de execução que seriam dificilmente identificadas em ambientes reais [1].

Através de testes de simulação, um processador pode ser comparado e avaliado quanto às quantidades de instruções executadas e seu tempo e velocidade. Esses dados possibilitam a comparação e análise dos desempenhos de diferentes processadores e permitem a fabricantes identificar possíveis necessidades de melhorias.

O presente trabalho foi desenvolvido com o objetivo de avaliar esses aspectos no processador PowerPC. Uma série de experimentos foi realizada sobre o processador utilizando o simulador MPSoCBench com as aplicações do benchmark ParMiBench. Os dados gerados foram em seguida analisados. Este relatório ilustra os resultados dos experimentos e das análises feitas pelo grupo.

Este trabalho está organizado conforme segue: A Seção 2 apresenta o processador, o simulador e as aplicações utilizadas nos experimentos; Na Seção 3 são mostrados os resultados dos experimentos e sua análise; As conclusões finais são apresentadas na Seção 4.

2. Fundamentação teórica

2.1. Processador PowerPC

O PowerPC (*Performance Optimization With Enhanced RISC – Performance Computing*) é um conjunto de instruções RISC criado em 1991, pelas empresas: IBM, Apple e Motorola, com o intuito de competir contra a Microsoft e Intel na área de computação pessoal.

O processador foi introduzido no mercado pela primeira vez em 1992, mas era um produto de muitos anos de pesquisa. É um derivado da série de microprocessadores Power da IBM, baseado em RISC. Foi muito reconhecido pela sua utilização na família de computadores Macintosh, porém também amplamente utilizado em diversas outras empresas, como a Sony, que utilizou o processador CELL, baseado totalmente no PowerPC, no console Playstation 3; e a Cisco, que utilizou processadores PowerPC embarcados em seus roteadores. A GPU Xenon, da Microsoft, também foi baseada no PowerPC, e utilizada no console Xbox 360. Da mesma forma fez a Nintendo, com seus processadores "Gekko" e "Broadway", utilizados nos consoles Gamecube e Wii. [2]

O PowerPC foi projetado de acordo com os princípios da arquitetura RISC, e permite implementações em superescala. Existem versões em 32 e 64 bits, baseadas na especificação básica da arquitetura Power, com as seguintes características de design [3]:

- Suporte para operacões em Little-endian e Big-endian;
- Formas de precisão simples e dupla para instruções de ponto flutuante;
- Especificação de 64 bits completa e compatível com o modo de 32 bits;
- Fusão entre multiplicação e adição ((a ← a + b * c));
- Arquitetura de gerenciamento de memória paginada, a qual é amplamente utilizada em sistemas de servidores e PCs;
- Arquitetura de gerenciamento de memória Book-E, substituindo a arquitetura de gerenciamento de memória paginada convencional para aplicações embarcadas.

2.2. Simulador MPSoCBench

Para a realização dos testes no PowerPC, foi escolhido o simulador MPSoCBench, uma ferramenta criada na Universidade de Campinas para atender diversos objetivos como: auxiliar a implementação e avaliação de novas ferramentas ou metodologias em designs de MPSoC; desenvolver e monitorar refinamentos de design para níveis de abstração baixos; e auxiliar a análise e otimização de novos componentes de hardware. [4]

O simulador fornece modelos para processadores ARM, SPARC, MIPS e PowerPC, que podem ser observados em plataformas de até 64 núcleos. Também são fornecidos cross-compiladores, IPs, interconexões e estimativas de energia para seus principais componentes: processadores, roteadores, memória principal e caches. [5]

2.3. Aplicações testadas

O simulador MPSoCBench fornece 17 aplicações paralelas, das quais sete pertencem ao benchmark ParMiBench, escolhido para a realização dos testes. As aplicações são descritas a seguir [5]:

- Basicmath: realiza cálculos matemáticos, como resolução de função cúbica, conversões de ângulos de graus para radianos e raízes quadradas. A paralelização é feita com particionamento de dados.
- Dijkstra: calcula todos os caminhos mínimos em um grafo representado por uma matriz de adjacências. Cada processador recebe um vértice e calcula os caminhos mínimos com origem nele.
- SHA: implementa o Secure Hash Algorithm, útil para gerar assinaturas digitais usadas em chaves criptográficas. Cada processador realiza o cálculo para um texto de um arquivo de entrada, em seguida todos guardam os seus dados gerados num único arquivo de saída.
- Stringsearch: encontra um padrão num conjunto de frases empregando algoritmos de comparação. A estratégia é particionar toda a coleção de padrões num número de subpadrões de acordo com o número de processadores utilizado.
- SusanCorners: um conjunto de funcionalidades de reconhecimento de imagens. É paralelizado com decomposição dos dados e é limitado para rodar em até 32 núcleos.
- SusanEdges: reconhece arestas numa imagem de entrada. É paralelizado com decomposição dos dados e é limitado para rodar em até 32 núcleos.
- SusanSmoothing: executa ajustes de limiar, brilho e suavidade de imagem. É paralelizado por decomposição de dados e é limitado para rodar em até 8 núcleos.

3. Experimentação

Nos experimentos foram computados dados de tempo e velocidade de simulação e quantidades de instruções executadas. Os dois primeiros aspectos são por padrão informados como saída do MPSoCBench. Para obter os números de instruções, o grupo criou variáveis contadoras no arquivo de código **proc_isa.cpp** do simulador. Cada variável foi responsável por contar as instruções de uma das sete categorias do PowerPC [6] implementadas no MPSoCBench, como detalhado seguir:

- aritmetica: responsável por contar todas as ocorrências de **instruções aritméticas**;
- logica: responsável por contar todas as ocorrências de **instruções lógicas**;
- desvio: responsável por contar todas as ocorrências de **instruções de desvio e comparação**;
- memoria: responsável por contar todas as ocorrências de **instruções de acesso à memória**, como *Store* e *Load*;
- rotateshift: responsável por contar todas as ocorrências de **instruções de rotação e deslocamento**;
- branchflow: responsável por contar todas as ocorrências de **instruções de *Branch* e *Flow*** e;
- processador: responsável por contar todas as ocorrências de **instruções de processador**.

3.1. Gráficos

Para melhor visualização dos resultados dos experimentos, foram montados gráficos com os dados gerados. Primeiramente, foram colocados em gráficos de barras as quantias totais de instruções executadas por cada aplicação. Os testes foram feitos utilizando 8 núcleos, os totais dos núcleos foram somados gerando os valores a seguir:

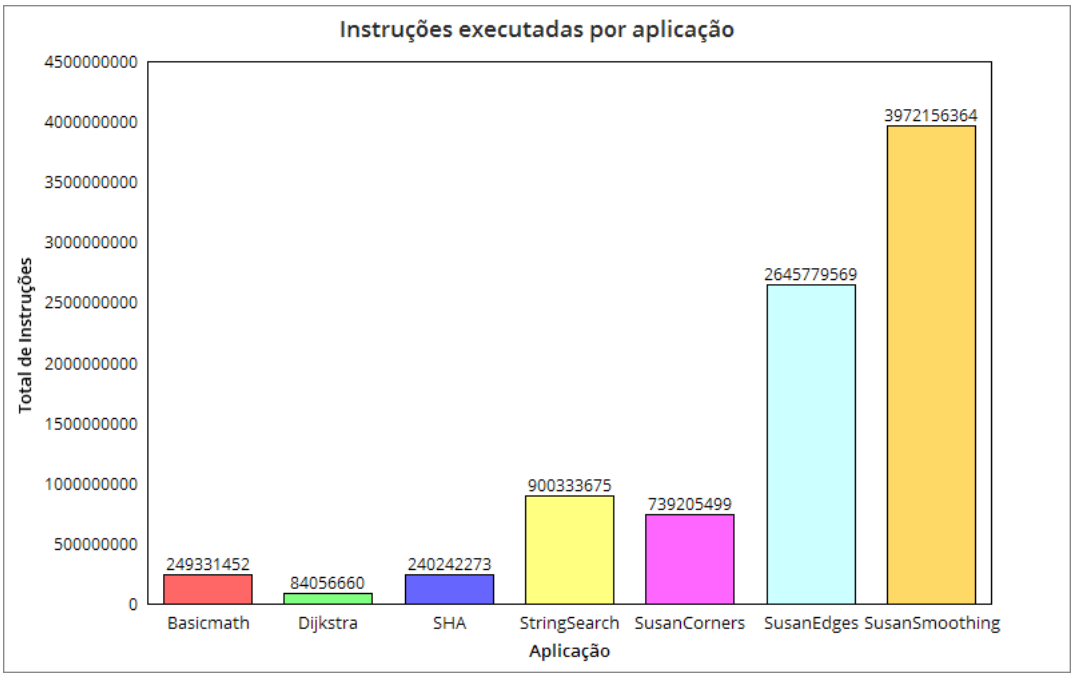

Figura 1. Total de instruções executadas por cada aplicação.

Em seguida, foi plotado um gráfico contendo os tempos gastos em cada simulação:

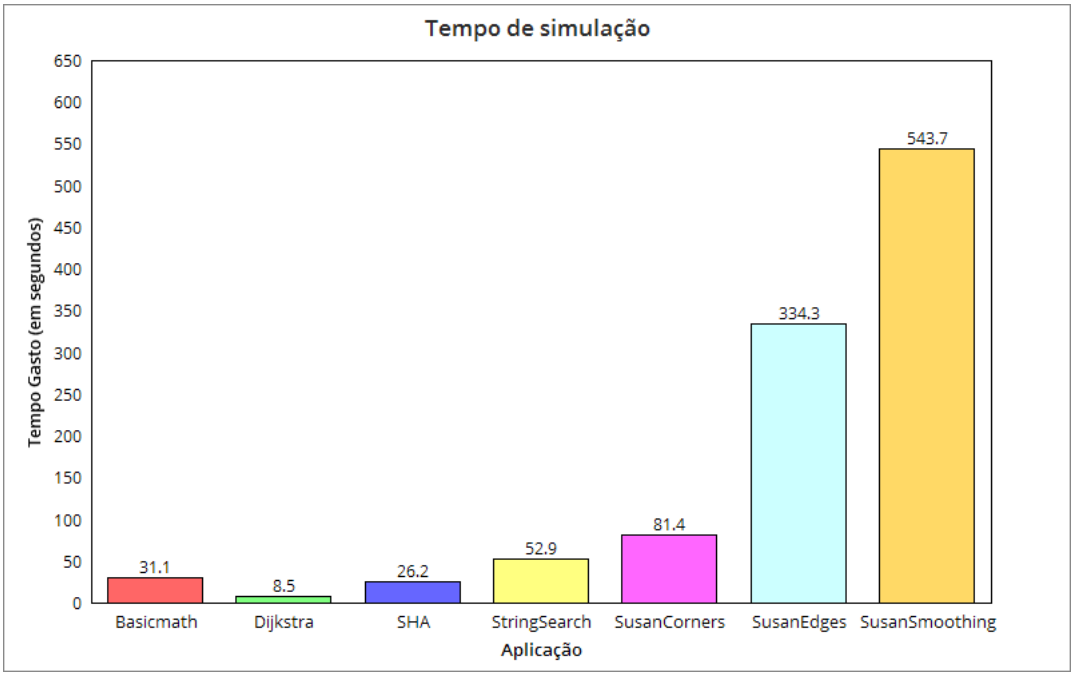

Figura 2. Total de tempo gasto na simulação de cada aplicação.

A relação entre número de instruções e tempo de execução é melhor visualizada no gráfico a seguir, o qual contém a velocidade de cada simulação:

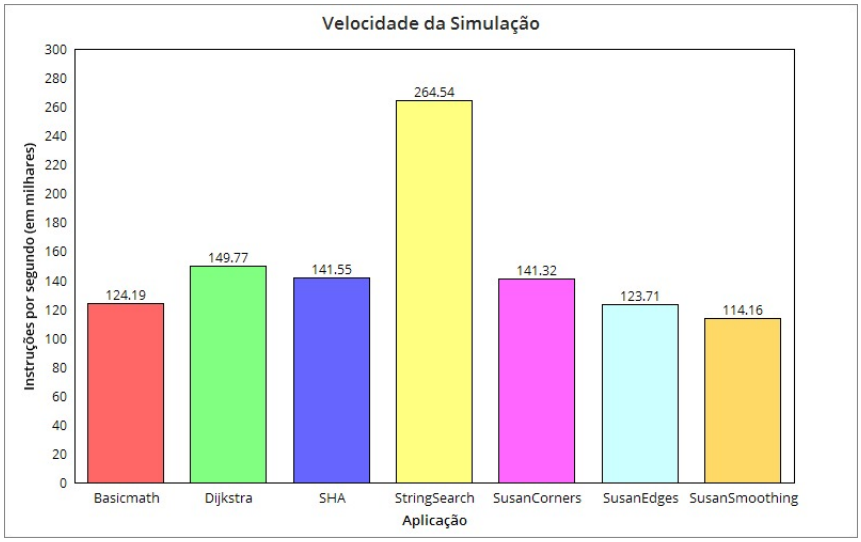

Figura 3. Velocidade de execução de cada aplicação.

O conjunto de instruções PowerPC é dividido em sete categorias implementadas no MPSoCBench: aritmética, lógica, desvio e comparação (no gráfico representada apenas pelo termo "Desvio"), memória, rotação e deslocamento (no gráfico, "Rotate"), *Branch* e processador.

Foram escolhidas quatro aplicações (Dijkstra, SHA, Strinsearch e SusanSmoothing) para terem seus resultados ilustrados de forma mais detalhada, comparando em gráficos suas quantidades de instruções executadas por cada categoria.

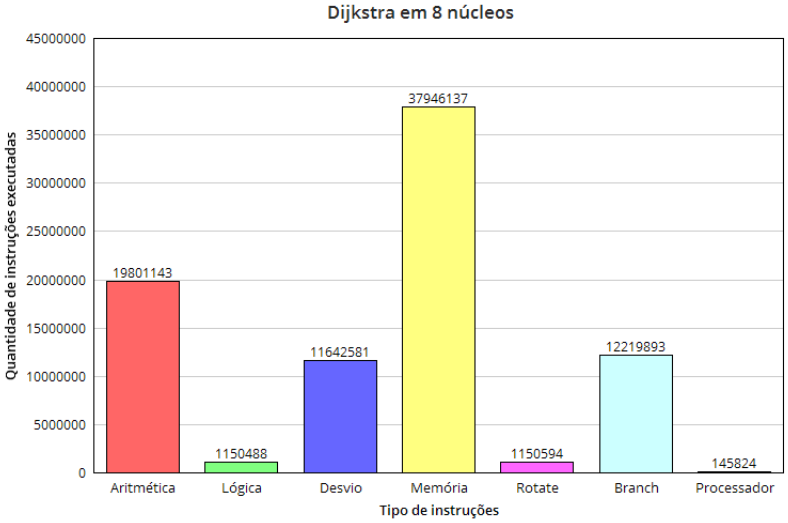

Figura 4. Quantidade de instruções por categoria da aplicação Dijkstra.

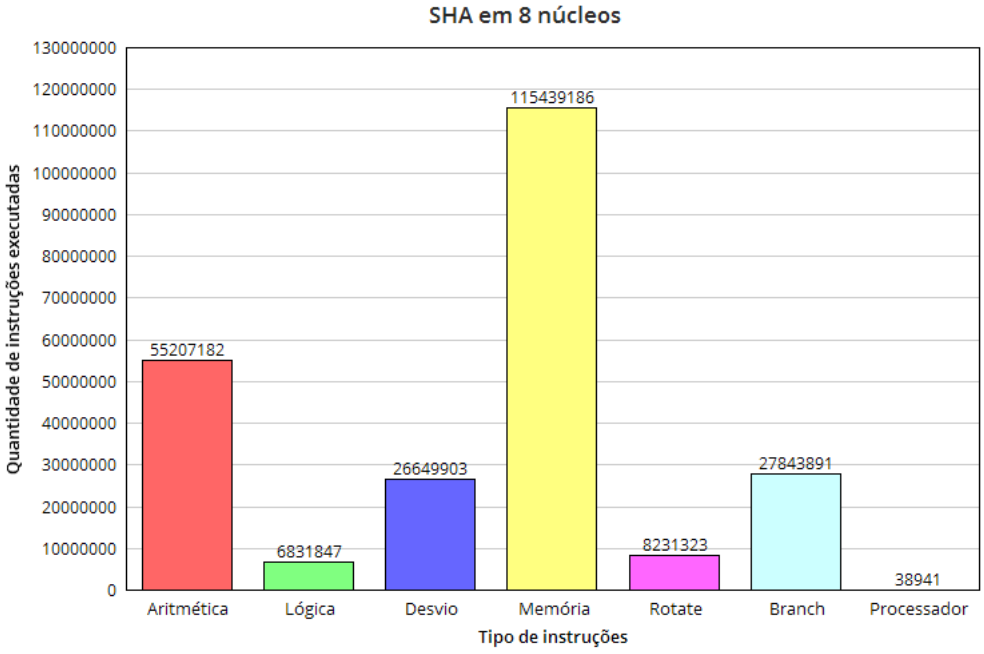

Figura 5. Quantidade de instruções por categoria da aplicação SHA.

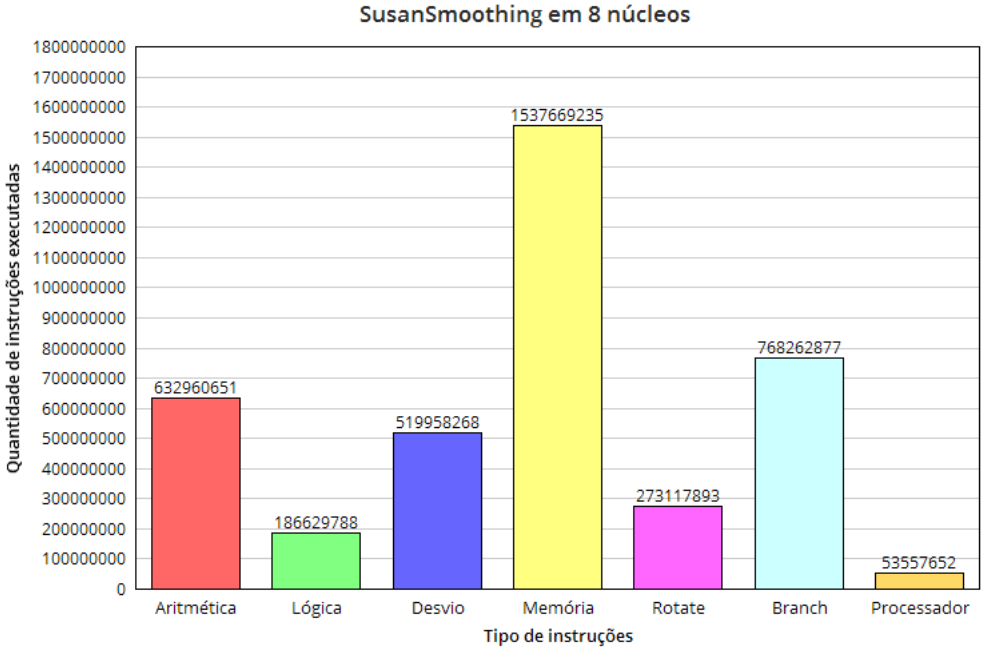

Figura 6. Quantidade de instruções por categoria da aplicação SusanSmoothing.

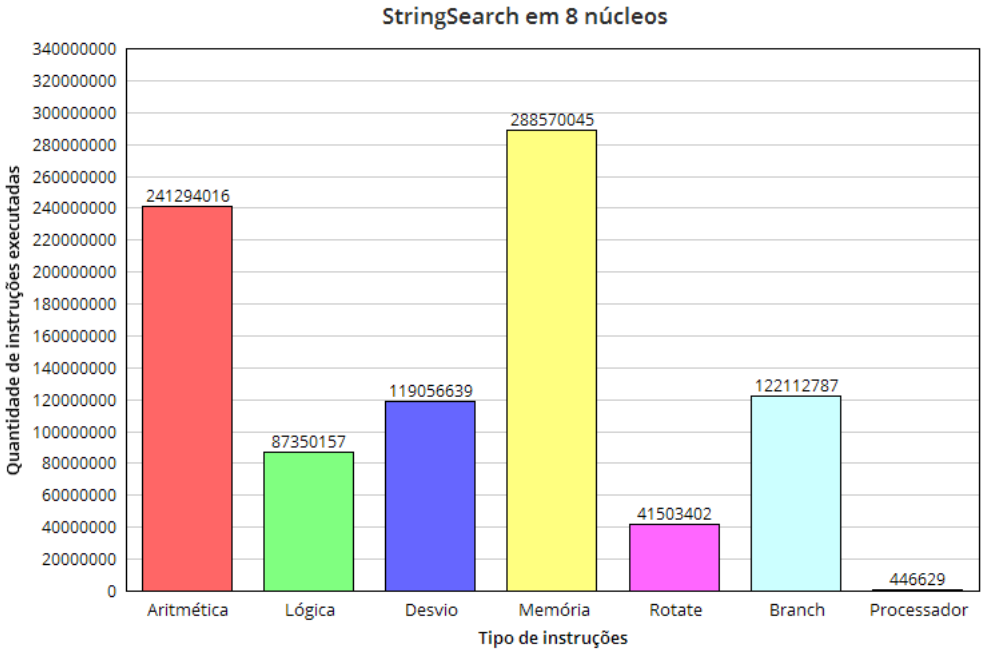

Figura 7. Quantidade de instruções por categoria da aplicação Stringsearch.

3.2. Análise dos resultados

Observando-se o gráfico Figura 1 apresentado na seção anterior, é possível notar que as aplicações Susan de trabalho com imagens, especialmente Edges e Smoothing, possuem uma quantidade de instruções muito maior que as demais testadas, gerando um contraste de mais de um bilhão de instruções com as aplicações Basicmath, SHA e Dijkstra. Como

já podia se esperar, o tempo de simulação, em Figura 2, gasto é proporcional ao total de instruções, sendo exceção apenas a aplicação Stringsearch que, apesar de possuir 200 milhões a mais de instruções que a SusanCorners, executa quase na metade do tempo desta.

O gráfico Figura 3 confirma essa conclusão, apontando que a velocidade em milhares de instruções por segundo de Stringsearch é quase duas vezes maior que a de todas as demais aplicações, as quais não possuem discrepância entre si nesse quesito.

A primeira aplicação escolhida para análise mais detalhada foi Dijkstra, Figura 4, a de menor tamanho e tempo e segunda maior velocidade. Foi percebido que Dijkstra, a qual soluciona um problema de teoria de grafos, executa em sua maioria instruções de memória e aritmética, além de desvios e *branches*. A quantia na demais categorias, lógica, rotação e processador, é irrisória.

Essa mesma distribuição foi observada na Figura 5, o gráfico para a aplicação SHA, a qual lida com um problema de criptografia.

A aplicação Susan, Figura 6, a de maior quantidade total de instruções em todo o ParMiBench, começa a alterar essa visão. Das quatro aplicações detalhadas, é a que possui mais instruções da categoria de processador, além de ter instruções de lógica e rotação numa proporção maior que Dijkstra e SHA. Também é notável a grande quantia de instruções *branch* que possui, maior que a de aritméticas.

A aplicação que rendeu a simulação mais rápida de todas, Stringsearch, Figura 7, mantém o padrão de muitas instruções de memória, porém com uma quantidade até então inédita de instruções aritméticas e lógicas. Por outro lado, ao contrário de SusanSmoothing, há poucas intruções de processador.

4. Conclusão

Através dos experimentos puderam ser destacadas algumas características do processador PowerPC, como, por exemplo, o fato de sua principal categoria de instruções ser a de memória, enquanto as instruções de processador, em geral, não são utilizadas em grande escala.

Durante os testes, o simulador MPSoCBench possibilitou visualizar, em detalhes, esse e outros aspectos, não abordados neste relatório, como estatísticas de uso de memória *cache*. Além das informações fornecidas como saída do programa, muitas outras podem ser obtidas através da exploração da característica de código aberto do simulador, a qual gera um leque de possibilidades para novos experimentos ao permitir a adição de variáveis contadoras.

Referências

[1] E. H. Cruz, J. H. Foleiss, G. P. Assunção, and R. A. Gonçalves, "Ferramenta de simulação de processador para ensino de graduação e pesquisa científica," *Anais SULCOMP*, vol. 4, 2008.

[2] K. J. Layer. (2009) Powerpc architecture. Disponível em: https://www.cs.uaf.edu/2009/fall/cs441/proj1/kevin/index.html .Acesso em: 10 Jun. 2016.

[3] IBM. Archived white papers. Disponível em: http://www-03.ibm.com/systems/power/resources/literature/archive.html .Acesso em: 10 Jun. 2016.

[4] A. Team. Mpsocbench. Disponível em: http://www.archc.org/benchs/mpsocbench/ .Acesso em: 10 Jun. 2016.

[5] L. D. D. Garanhani, "Mpsocbench: a framework for high-level evaluation of multiprocessor system-on-chip tools and methodologies," 2015.

[6] F. Semiconductor, "Programming environments manual for 32-bit implementations of the powerpctm architecture," 2005.