Técnicas para Avaliação do Desempenho de Arquiteturas Super Escalares

Gabriel Pereira da Silva[@]; Edil S. T. Fernandes*

([@]NCE e COPPE-Sistemas) (*COPPE-Sistemas)
Universidade Federal do Rio de Janeiro
Cx. Postal 68511 - CEP 21945-970
Rio de Janeiro - RJ
e-mail:gabriel@cos.ufrj.br; edil@cos.ufrj.br

Resumo

Durante o projeto de um processador é necessário realizar inúmeras simulações para determinar as melhores opções que serão incorporadas na sua arquitetura. *Trace-driven simulation* é uma das técnicas mais utilizadas. O *trace* obtido é processado pelo simulador do modelo de arquitetura em teste e a partir daí são extraídas estatísticas de desempenho do modelo. Para cada modificação na arquitetura investigada, precisamos simular novamente o mesmo conjunto de programas de teste. Neste artigo é apresentado um método alternativo de avaliação que usa diversas amostras, igualmente espaçadas ao longo da execução do programa, para formar um *trace* reduzido, porém representativo, da execução completa de cada programa de teste. Este *trace* amostrado é utilizado como entrada pelo simulador de uma arquitetura super escalar, permitindo que as modificações sejam avaliadas em um tempo muito mais reduzido. Experimentos com programas de teste mostraram que com apenas 1% das instruções é possível reproduzir o comportamento da arquitetura super escalar com o *trace* completo.

Abstract

Trace-driven simulation is a powerful technique to help designers during the specification of new machines. The technique allows to reproduce the machine behavior even before its actual implementation. For this reason, it plays an important role to assess the performance of new superscalar processors. In order to achieve a higher degree of parallel activities, the architecture of these very sophisticated processors must be very well balanced, and the evaluation of the performance impact caused by each modification in the basic architecture model requires the simulation of the overall set of test programs: a very time consuming task because each benchmark program consists of billions of instructions demanding a very large simulation time. In this paper, we present an alternative evaluation method that uses several samples, evenly distributed in time, to compose a reduced but representative trace from the complete benchmark execution. This sampled trace is used as input to the architecture model, and it allows a faster assessment of the effect caused by each modification. Experiments with test programs have shown that only with 1% of the instructions is possible to reproduce the behavior of the superscalar architecture with the complete trace.

1 Introdução

No projeto da arquitetura de um processador é imprescindível a realização de simulações para determinar as melhores opções a serem utilizadas na sua arquitetura. Uma das técnicas mais utilizadas é a de *trace-driven simulation*. Neste método, é gerado um *trace* dinâmico dos programas de avaliação, que contém uma seqüência dinâmica dos endereços e códigos das instruções, e dos endereços dos dados que ocorrem durante a execução do programa. Este *trace* é aplicado ao modelo de arquitetura em teste, e a partir daí são levantadas estatísticas de desempenho do modelo. É necessária uma nova execução de todo o conjunto de programas de avaliação para cada modificação na arquitetura a ser avaliada.

A necessidade de melhor avaliar o desempenho dos microprocessadores, levou a elaboração de sofisticadas suítes de programas (p.ex. SPEC92), de forma a representar uma aplicação típica de usuário. Estas suítes executam alguns bilhões de instruções, representando um problema para o projetista, quando avaliando as opções de projeto de sua arquitetura. A execução completa da suíte SPEC92 requer mais de 100 bilhões de instruções para cada modelo de arquitetura. Com velocidades de simulação na ordem de dezenas de milhares de instruções por segundo, a simulação completa da suíte de avaliação levaria vários dias.

Além dos tempos envolvidos, uma outra questão importante é a quantidade de espaço em disco gasta para armazenar os *traces*. Um programa com 1 bilhão de instruções, considerando-se apenas o espaço gasto para armazenar as referências de busca das instruções, precisaria de 4 Gbytes de espaço de armazenamento. Algumas técnicas para redução do espaço gasto podem ser aplicadas [PLE94], mas o espaço gasto ainda é considerável.

Para tornar esta tarefa viável, propomos neste trabalho a utilização de técnicas de amostragem para obtenção dos *traces*. O método de amostragem proposto neste trabalho é o de utilizarmos diversas amostras, igualmente espaçadas ao longo da execução do programa, para formar um conjunto representativo do programa de avaliação. Esta técnica foi utilizada na avaliação de diversos processadores super escalares comerciais [SHI94, LAU93], com variações que serão detalhadas ao longo deste trabalho. Apresentaremos a seguir os resultados deste trabalho, realizado para otimizar os tempos de execução e espaço de armazenamento gastos na realização de estudos de arquitetura de processadores no programa de Engenharia de Sistemas da COPPE/UFRJ.

2 Trabalhos Anteriores

O uso de técnicas de amostragem se aplica tanto para a avaliação do comportamento de *caches* [LAH88, KES91, LIU93], como o de modelos de arquitetura de processadores [LAU93,SHI94].

Esta técnica foi utilizada também no desenvolvimento de um monitor de desempenho [MAR93]. Nesse desenvolvimento, o total de amostras utilizado correspondeu a 10% do total de instruções, com cerca de 20 amostras de 0.5 MB cada. Quando as *caches* simuladas possuíam tamanhos de 16 KB e 128KB, o erro absoluto na taxa de falhas da *cache* não excedia a 0.3%. Para tamanhos maiores de *cache* (aprox. 1 MB), o uso de amostras maiores (4 a 8 milhões de referências cada) permite melhorar a acurácia, mantendo-se a mesma taxa de amostragem (10%).

Outros trabalhos [LAH88] indicam que 35 amostras são suficientes para caracterizar bem (erro percentual menor que 5,5%) a taxa de falhas para aplicações LISP, quando a taxa de amostragem se situa entre 3 e 10%. Em [MAR93] encontramos curvas mostrando que, mantendo-se o mesmo percentual de 10%, a partir de 10 amostras o erro em relação a taxa de falhas real é menor que 5% para a maioria dos programas. A exceção ocorre nos programas onde a taxa de erro absoluta é muito pequena (p.ex., espresso).

Técnicas de amostragem foram também utilizadas no desenvolvimento de processadores comerciais. No PowerPC [SHI94], por exemplo, as técnicas utilizadas são bastante simples, consistindo de 200 amostras de 5000 instruções contínuas, distribuídas uniformemente ao longo do tempo de execução, totalizando 1.000.000 de instruções, independente do tamanho e do tipo do programa em estudo. Este procedimento não é recomendável, já que o percentual de instruções amostrado variou com cada programa. Para alguns programas de teste o tamanho do *trace* amostrado foi pequeno e o tamanho de cada amostra comprometeu a acurácia das taxas de falha das *caches*, provocando alguns desvios em relação ao *trace* completo.

Nesse trabalho [SHI94], embora o erro da média geométrica do número de Instruções executadas Por Ciclo (IPC) estivesse dentro de 2%, a margem de erro para os *traces*, individualmente, chegou a 13% como no programa sc. Em outro tipo de programa utilizado nesta mesma avaliação (de nome TPC), a influência do estado da *cache* no início de cada amostra foi tão grande, que o IPC amostrado se afastou de maneira inaceitável do IPC obtido com o *trace* completo. O uso de amostras maiores, aumentando o percentual de amostragem, poderia reduzir estas distorções.

Em outro trabalho analisado [LAU93] foram realizadas amostragens menores que 0.5% do *trace* completo, significando tempos de simulação 200 vezes menores. Nesse caso, o número inicial de amostras foi de apenas 15, cada uma delas com

250.000 instruções uniformemente distribuídas ao longo do *trace* completo. Caso as amostras tomadas não fossem representativas do comportamento do programa, novas amostras eram então agregadas até obter um conjunto representativo de amostras do programa. Nas amostras não foram incluídas instruções do espaço do sistema (chamadas ao sistema, *traps*, interrupções, etc.).

Existem várias medidas para determinar quando o conjunto de amostras é representativo. Em [LAU93] a verificação foi dividida em duas etapas: verificações rápidas para determinar se o conjunto tomado é claramente insuficiente, e outras mais elaborados para uma verificação final no conjunto de amostras. As verificações mais rápidas usam a freqüência das instruções e das funções e estatísticas de *cache*, além do IPC, para determinar quando o conjunto de amostras é suficiente. A verificação mais elaborada compara o desempenho de um conjunto de micro-arquiteturas obtidos pelo *trace* amostrado com o obtido pelo completo. Diferenças sempre menores que 0,5% na freqüência de execução das instruções foram obtidas, por exemplo, para o programa gcc. O estado inicial da memória *cache* foi reproduzido no início de cada amostra, tornando a diferença absoluta da taxa de falhas sempre menor que 2% entre o *trace* amostrado e o completo. A diferença entre o IPC do SPECint92 para os dois *traces* não foi maior que 3% para os diferentes tipos de arquiteturas simuladas.

3 Metodologia

Nosso objetivo principal consiste em obter *traces* para avaliar com confiabilidade e mais rapidamente diferentes configurações super escalares. Neste artigo apresentamos quando e como devem ser obtidos estes *traces* para que os objetivos sejam atingidos.

Utilizamos em nosso ambiente de teste um simulador super escalar parametrizável com conjunto de instruções compatível com o i860 [INT91] e que utiliza o algoritmo de Tomasulo [TOM67] para resolução dinâmica de dependências. O simulador, denominado *tomasulo* [HAO93], reproduz o comportamento de diversas configurações derivadas do modelo básico ao modificarmos um dos parâmetros arquiteturais (número de unidades funcionais, estações de reserva, número de instruções despachadas simultaneamente, tamanho da *cache* de dados e instruções, tamanho da *cache* de endereços de desvio (BTB) [LEE84], entre outros). Como resultado o simulador fornece o número de ciclos gastos na execução do programa, o *speed-up* em relação a arquiteturas convencionais, taxas de acerto nas *caches* e no BTB, taxas de ocupação das unidades funcionais e estações de reserva, entre outros.

A questão levantada é se poderíamos realizar comparações válidas entre diferentes configurações de arquitetura com o uso da técnica *trace* amostrado.

Para a geração dos *traces*, o simulador sim860a [FER93], desenvolvido para o ambiente DOS, foi adaptado para um ambiente UNIX, e rotinas emulando chamadas ao sistema operacional foram acrescentadas. Outra modificação introduzida consistiu na introdução de parâmetros indicando o número e a taxa de amostragem.

O simulador sim860a não anota as instruções executadas durante as chamadas ao sistema operacional, já que as mesmas são emuladas pelo processador/sistema operacional hospedeiro. Avaliando a arquitetura Alpha [CVE94] com os programas SPEC92, verificou-se que o percentual de tempo gasto em chamadas ao sistema operacional é menor que 3% para a maior parte dos programas. A exceção é o programa *compress*, que por utilizar *pipes*, gasta perto de 15% do tempo em chamadas ao sistema operacional, não sendo por isto utilizado em nossa análise.

A saída do simulador sim860a pode ser redirecionada para um programa compactador e, para cada 1.000.000 de instruções simuladas, obtém-se em média um arquivo com 300 Kbytes, ou seja, uma redução de espaço de de 1:200. Os arquivos compactados são enviados para um descompactador e daí, via *pipe*, para o simulador *tomasulo*. Assim podemos manter os *traces* sempre compactados, com grande economia de espaço em disco.

Podemos também executar simulações em paralelo em diversas máquinas, com grande economia de tempo. No estudo realizado, simulamos programas com até 100 milhões de instruções.

Considerando-se que o modelo de arquitetura contém somente uma *cache* (para dados e instruções) com 8 Kbytes, não utilizamos nenhuma forma de previsão do estado das *caches*, no início de cada amostra.

Também desenvolvemos um programa para verificar a distribuição das instruções, individualmente e por tipo de unidade funcional, em cada um dos *traces*, para verificar a sua representatividade.

4 A Escolha da Amostras

Analisamos quatro programas: os inteiros *espresso* e *xlisp*; *whetstone* e *tomcatv* de ponto flutuante. Analisamos *traces* incluindo 1% e 100% das instruções de cada programa. O número de amostras variou entre uma dezena e algumas centenas de amostras, dependendo das características do programa de teste. A Tabela 1 lista o número de amostras para cada caso.

As amostras são formadas pelas instruções iniciais de cada intervalo de amostragem e contêm 100.000 ou 1.000.000 de instruções, de acordo com o tamanho do programa, mas sempre no percentual de 1% do total de instruções. As entradas de dados do SPEC92 foram alteradas, para permitir tempos de execução e tamanhos

Programa	Total de Instruções	Tamanho das Amostras	Número de Amostras
Espresso	1.230.000	1.000	12
Whetstone	2.190.000	1.000	22
Xlisp	15.500.000	1.000	160
Tomcatv	75.000.000	100.000	75

Tabela 1: Programas de Teste

	MÁQUINA A					MÁQUINA B				
	Soma	Mult	Graf	Memo	Core	Soma	Mult	Graf	Memo	Core
FU	2	2	1	1	1	2	2	1	1	3
RS	4	4	2	2	2	10	10	5	5	15

Tabela 2: Configurações de Arquitetura

de *traces* viáveis.

Para cada um dos programas, produzimos gráficos mostrando a freqüência de execução de cada instrução, tanto no *trace* amostrado, como no *trace* completo.

Como queremos validar o uso da técnica de amostragem para estudos comparativos entre diferentes arquiteturas, os dois tipos de *trace* foram submetidos ao simulador *tomasulo* com as configurações de arquitetura A e B, descritas na Tabela 2.

Os resultados da simulação de cada arquitetura com os dois *traces* são sumarizados nas tabelas a seguir. Os dados mais significativos são o total de ciclos gastos na execução, taxa de acertos na *cache* de endereços de desvio (BTB), *speed-up* em relação a arquiteturas convencionais, e a taxa de ocupação das unidades funcionais, especificadas por tipo e número. O *speed-up* apresentado refere-se ao IPC da arquitetura super escalar dividido pelo IPC de uma arquitetura seqüencial.

5 Análise dos Resultados

5.1 Espresso

O programa espresso é um dos menores programas simulados. Ele executa 1.230.000 instruções e o *trace* amostrado possui 12 amostras de 1.000 instruções. As Tabelas 3 e 4 mostram que é um programa com elevadas taxas de acerto no (BTB), e as variações no *speed-up* indicam que é sensível a mudanças nas taxas de amostragem. O pequeno tamanho das amostras afeta diretamente a taxa de acerto nas duas *caches* (instruções/dados e BTB), aumentando a taxa de falhas no BTB

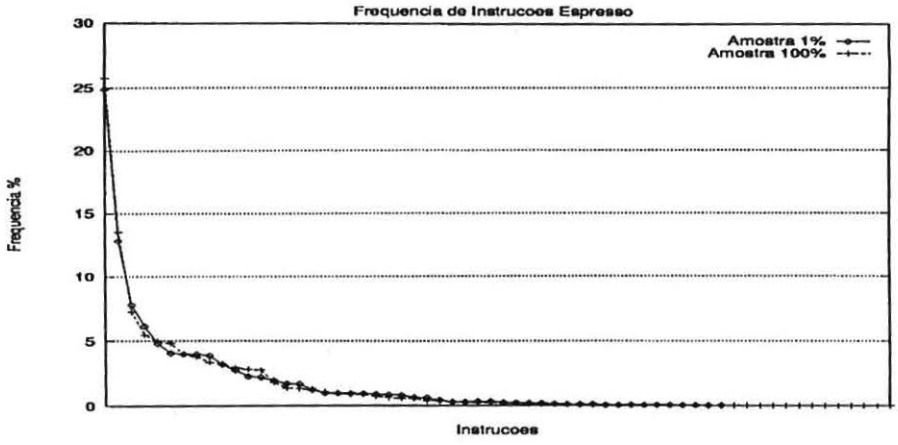

Figura 1: Freqüência das Instruções Espresso

	MÁQUINA A	MÁQUINA B
Tempo total:	41.706 ciclos	37.886 ciclos
Speed-Up:	2,00	2,20
Hit no BTB :	77,21 %	77.21 %

Tabela 3: Estatísticas Espresso 1%

	MÁQUINA A	MÁQUINA B
Tempo total:	3.745.460 ciclos	3.365.880 ciclos
Speed-Up:	2,10	2,34
Hit no BTB :	92.08%	92.08%

Tabela 4: Estatísticas Espresso 100%

	MÁQUINA A			MÁQUINA B		
	F(1)	F(2)	F(3)	F(1)	F(2)	F(3)
Soma PF	0.23%	0.26%		0.32%	0.22%	
Mult PF	1.27%	0.30%		1.12%	0.62%	
Gráfica	0.73%			0.81%		
Memória	79.11%			87.08%		
Core	21.13%			9.71%	7.69%	5.86%

Tabela 5: Ocupação das Unidades Funcionais (Espresso 1%)

	MÁQUINA A			MÁQUINA B		
	F(1)	F(2)	F(3)	F(1)	F(2)	F(3)
Soma	0.15%	0.16%		0.19%	0.16%	
Mult	0.53%	0.16%		0.56%	0.21%	
Gráfica	0.27%			0.30%		
Memória	79.30%			88.24%		
Core	22.60%			10.47%	8.21%	6.46%

Tabela 6: Ocupação das Unidades Funcionais (Espresso 100%)

e reduzindo o *speed-up* absoluto. Mesmo assim, as diferenças relativas entre as arquiteturas A e B são mantidas (10% no *trace* amostrado e 11,4% no completo), o que ainda permite obter dados comparativos válidos.

Nas Tabelas 5 e 6, verificamos que a diferença entre a taxa de distribuição das instruções por unidade funcional não é significativa (não é maior que 1,5%) , em valores absolutos.

A Figura 1 mostra a freqüência das instruções dos *traces* amostrado e completo provenientes do programa Espresso. Conforme podemos ver, ocorre a mudança no valor da freqüência de algumas instruções de um *trace* para outro.

Embora o tamanho do *trace* seja bastante eficiente em termos do tempo de simulação, o pequeno número e o reduzido tamanho das amostras provocou um desvio na freqüência de instruções nos dois *traces*, ocasionando a maior diferença (cerca de 6%) nos dois *speed-ups* de todos os experimentos realizados (vide Tabelas 3 e 4).

5.2 Whetstone

A Figura 2 mostra a distribuição de instruções nos dois *traces* do programa Whetstone. Embora possamos observar pequenas diferenças entre as freqüências das instruções, isso não compromete a qualidade da amostra.

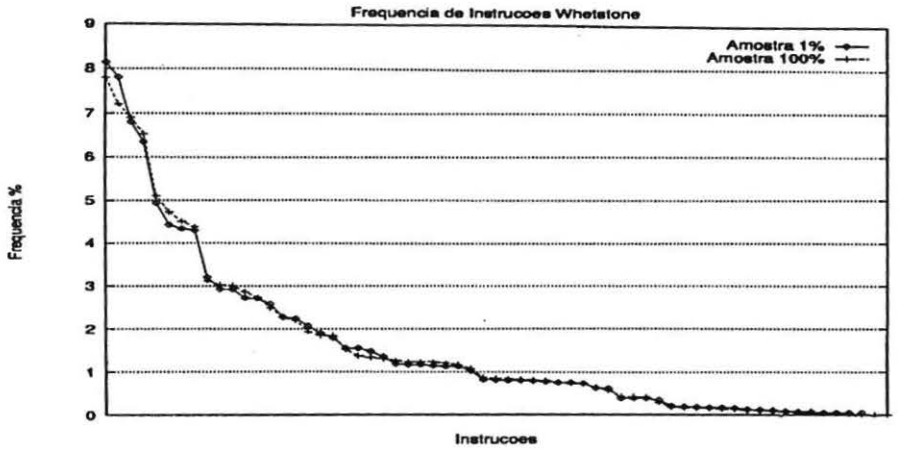

Figura 2: Freqüência das Instruções Whetstone

	MÁQUINA A	MÁQUINA B
Tempo total:	58.866 ciclos	48.351 ciclos
Speed-Up:	2,52	3,06
Hit na BTB :	92,08 %	92,08 %

Tabela 7: Estatísticas Whetstone 1%

	MÁQUINA A	MÁQUINA B
Tempo total:	5.741.995 ciclos	4.691.716 ciclos
Speed-Up:	2,57	3,14
Hit na BTB :	99,54 %	99,54%

Tabela 8: Estatísticas Whetstone 100%

	MÁQUINA A			MÁQUINA B		
	F(1)	F(2)	F(3)	F(1)	F(2)	F(3)
Soma PF	12,39%	6,25%		13,71%	8,98%	
Mult PF	15,72%	11,36%		19,02%	13,96%	
Gráfica	15,57%			18,96%		
Memória	52,99%			64,52%		
Core	22,50%			12,83%	8,89%	5,67%

Tabela 9: Ocupação das Unidades Funcionais (Whetstone 1%)

Quando avaliamos a execução do *trace* amostrado nas arquiteturas A e B (Tabelas 7 e 8), podemos observar que existe uma diferença de cerca de 7,5% na taxa de acerto no BTB, em relação ao *trace* completo. Isto implica na pequena diferença entre o *speed-up* real e o amostrado (cerca de 2,6%), mas que afeta igualmente as duas arquiteturas, que apresentam valores comparativos muito próximos (21% e 22% de melhoria da arquitetura B sobre a A, para os *traces* amostrado e completo, respectivamente).

Em termos da ocupação das unidades funcionais (Tabelas 9 e 10), os valores encontrados apresentam diferenças absolutas de no máximo 1,75% (8% em termos relativos) o que é bastante razoável.

O *trace* amostrado do Whetstone é composto por 22 amostras, cada uma com 1.000 instruções. O número de amostras é aceitável, mas poderia ser melhor. A taxa de acerto no BTB foi também prejudicada pelo pequeno tamanho da amostra em relação ao tamanho da *cache*.

5.3 Xlisp

A distribuição de instruções dos *traces* produzidos a partir do programa xlisp (vide Figura 3) apresenta uma diferença inferior a 3%. Além disso, não há alterações de ordem no histograma para as instruções mais executadas do programa (88%).

Examinando as Tabelas 11 e 12, podemos observar pequenas diferenças de desempenho provocadas pelos dois *traces* em cada arquitetura (configurações A e B). Contudo, quando comparamos as duas configurações, podemos verificar que elas apresentam a mesma taxa de falhas no BTB e *speed-up* relativo bastante próximo (1,09 versus 1,095).

Quanto ao nível de ocupação das unidades funcionais, os valores encontrados são bem próximos, resultado talvez de uma distribuição equivalente das instruções, para os dois *traces* (vide Tabelas 13 e 14).

O *trace* do programa xlisp contém 160 amostras, cada uma com 1.000 instruções.

	MÁQUINA A			MÁQUINA B		
	F(1)	F(2)	F(3)	F(1)	F(2)	F(3)
Soma PF	13,53%	7,04%		15,17%	10,01%	
Mult PF	17,16%	12,30%		20,76%	15,30%	
Gráfica	16,50%			20,20%		
Memória	52,00%			63,65%		
Core	22,33%			12,82%	8,91%	5,60%

Tabela 10: Ocupação das Unidades Funcionais (Whestone 100%)

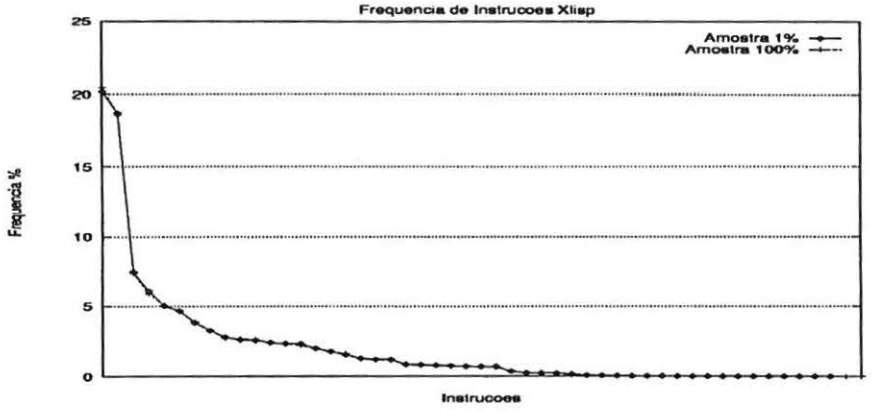

Figura 3: Freqüência das Instruções Xlisp

	MÁQUINA A	MÁQUINA B
Tempo total:	462.233 ciclos	425.144 ciclos
Speed-Up:	2,11	2,29
Hit no BTB :	95,54%	95.54%

Tabela 11: Estatísticas Xlisp 1%

	MÁQUINA A	MÁQUINA B
Tempo total:	44.087.642 ciclos	40.329.003 ciclos
Speed-Up:	2,20	2,41
Hit no BTB :	97,86%	97,86%

Tabela 12: Estatísticas Xlisp 100%

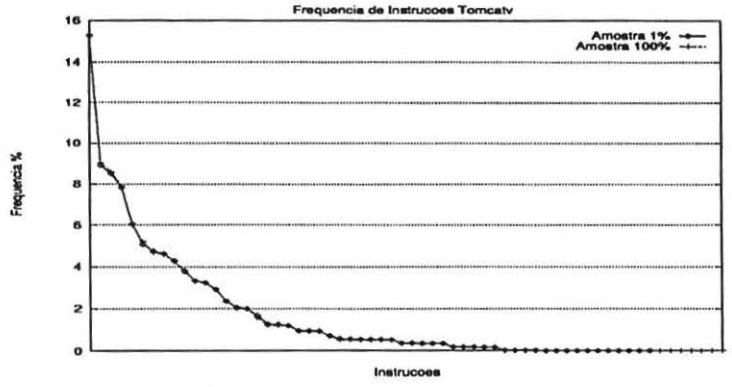

Figura 4: Freqüência das Instruções Tomcatv

Conforme ilustrado na Figura 3, o grande número de amostras permitiu caracterizar bem o programa, em relação a freqüência de execução de cada instrução.

Como anteriormente, o pequeno tamanho das amostras influenciou a taxa de acerto do BTB e da *cache* (ocorreu uma pequena redução nas taxas de acerto). Talvez isto explique o *speed-up* ligeiramente menor do *trace* amostrado em relação ao *trace* completo.

Um dos objetivos propostos é o de estudar o impacto do uso do *trace* amostrado na avaliação de mudanças arquiteturais. Neste sentido, a melhoria de desempenho que a arquitetura B apresenta sobre a arquitetura A é quase a mesma, utilizando o *trace* amostrado (9%) ou completo (9,05%).

5.4 Tomcatv

No programa tomcatv, a maior diferença absoluta entre as instruções do *trace* amostrado e do completo é inferior a 0,14%; e a maior diferença relativa não é superior a 2,5% para as instruções que somam 80% do total de instruções executadas no programa. O resultado pode ser visto na Figura 4.

Quando comparamos as arquiteturas A e B (vide Tabelas 15 e 16), podemos observar que os dois *traces* apresentam as mesmas diferenças de *speed-up* e de taxa de acerto no BTB. A arquitetura B apresentou um desempenho que é 12,7% superior ao da arquitetura A para os dois *traces*, e as taxas de acerto no BTB das duas configurações são idênticas quando do processamento do mesmo *trace*.

Conforme listado nas Tabelas 17 e 18, as taxas de ocupação das unidades funcionais de cada máquina são muito semelhantes para cada *trace*.

O *trace* do programa Tomcatv contém 75 amostras, cada uma com 100.000 instruções. Estas são as condições ideais para a obtenção de um *trace* amostrado, já

	MÁQUINA A			MÁQUINA B		
	F(1)	F(2)	F(3)	F(1)	F(2)	F(3)
Soma PF	0,02%	0,02%		0,02%	0,02%	
Mult PF	0,03%	0,02%		0,03%	0,03%	
Gráfica	0,01%			0,01%		
Memória	86,53%			94,07%		
Core	26,78%			11,98%	9,72%	7,42%

Tabela 13: Ocupação das Unidades Funcionais (Xlisp 1%)

	MÁQUINA A			MÁQUINA B		
	F(1)	F(2)	F(3)	F(1)	F(2)	F(3)
Soma PF	0,04%	0,05%		0,05%	0,04%	
Mult PF	0,07%	0,05%		0,07%	0,02%	
Gráfica	0,02%			0,02%		
Memória	86,17%			94,20%		
Core	27,94%			12,47%	10,29%	7,79%

Tabela 14: Ocupação das Unidades Funcionais (Xlisp 100%)

	MÁQUINA A	MÁQUINA B
Tempo total:	1.605.432ciclos	1.423.473 ciclos
Speed-Up:	2,99	3,37
Hit no BTB :	99,82 %	99,82 %

Tabela 15: Estatísticas Tomcatv 1%

	MÁQUINA A	MÁQUINA B
Tempo total:	158.943.005 ciclos	141.014.855 ciclos
Speed-Up:	3,01	3,39
Hit no BTB :	100.00 %	100.00 %

Tabela 16: Estatísticas Tomcatv 100%

	MÁQUINA A			MÁQUINA B		
	F(1)	F(2)	F(3)	F(1)	F(2)	F(3)
Soma PF	17,6%	9,5%		16,6%	13,9%	
Mult PF	19,2%	11,5%		20,0%	14,6%	
Gráfica	9,2%			10,3%		
Memória	74,7%			84,3%		
Core	27,0%			13,6%	9,9%	7,0%

Tabela 17: Ocupação das Unidades Funcionais (Tomcatv 1%)

	MÁQUINA A			MÁQUINA B		
	F(1)	F(2)	F(3)	F(1)	F(2)	F(3)
Soma PF	17,6%	9,4%		16,5%	14,0%	
Mult PF	19,7%	11,9%		20,5%	15,1%	
Gráfica	9,1%			10,2%		
Memória	74,4%			83,9%		
Core	27,3%			13,7%	9,9%	7,1%

Tabela 18: Ocupação das Unidades Funcionais (Tomcatv 100%)

que o número de amostras é bastante acima daquele considerado ideal (em torno de 35), e o tamanho dos *traces* é consideravelmente maior que o tamanho da *cache* (400KB x 8KB), o que reduz as variações na taxa de acerto na *cache* e no BTB.

O tamanho deste *trace* (i.e., número e tamanho de amostras) associado a uma distribuição representativa de instruções, permite substituir o *trace* completo pelo amostrado, sem qualquer perda de qualidade nos resultados.

6 Conclusões

Intuitivamente a eficiência do *trace* amostrado é justificada pela grande diferença entre os tamanhos do código objeto e do *trace* completo de cada programa. Isto indica a presença de *loops* de instruções operando sobre vários conjuntos de dados.

A avaliação de uma arquitetura com *trace* amostrado será bem sucedida se este possuir características similares ao *trace* completo. Trabalhos anteriores sugerem o uso de diversos indicadores [MAR93] para caracterizar a qualidade do *trace* amostrado. Dentre esses, escolhemos em nosso trabalho o uso do IPC e da freqüência de execução das instruções.

Um número maior de amostras permite obter freqüências de instruções bem próximas nos dois tipos de *traces*. Caso necessário, o número de amostras pode ser aumentado até que as freqüências obtidas sejam semelhantes. Na escolha do tamanho de cada amostra deve-se levar em conta o tamanho das *caches* simuladas. O tamanho das amostras pode ser dilatado, se o tamanho das *caches* simuladas for grande e se o IPC amostrado divergir em demasia. Caso o tamanho das amostras aumente proibitivamente, recomendamos como alternativa a reconstituição do estado da *cache* no início de cada amostra.

As dificuldades encontradas na geração e armazenamento dos *traces* completos, que consomem longos tempos de simulação e espaço em disco, limitaram o número de experimentos. Apesar disto, os resultados apresentados validam a proposta do uso de *trace* amostrado para comparar diferentes configurações super escalares.

Quando da avaliação do desempenho (*speed-up*) de uma classe de arquiteturas super escalares, a utilização de *traces* com 1% do tamanho original, provocou reduções nos tempos de simulação superiores a 100 vezes, com erros inferiores a 6,5%. Examinando a taxa de ocupação das unidades funcionais, verificamos que elas apresentaram erros aceitáveis (erros absolutos menores que 1,5%), garantindo desse modo uma avaliação segura da arquitetura em teste.

A redução nos tempos de simulação e de espaço em disco, eleva significativamente o número de alternativas que podem ser avaliadas pelo projetista da arquitetura super escalar.

7 Bibliografia

[CVE94] Cvetanovic, Z. and Bhandarker, D. "Characterization of Alpha AXP Performance Using TP and SPEC Workloads", Proceedings of the ISCA94, pp. 60-70, April 1994.

[INT91] Intel, "i860 XR 64-bit Microprocessor", Intel Corp., June 1991, 78pp.

[FER93] Souza, Alberto F. "Avaliando Parâmetros de uma Arquitetura VLIW", Tese de Mestrado, COPPE/Sistemas, Abril de 1993.

[HAO93] Hao, Hsing T. "Efeito da Prediçao de Desvios e da Interrupção Precisa no Desempenho de Processadores Super Escalares", COPPE/UFRJ, Julho 1993.

[KES91] Kessler, R.E.; Hill, M.D. ; Wood, D.A. "A Comparision of Trace Sampling Techniques for Multi-Megabytes Caches" , Technical Report 1048, University of Wisconsin, Computer Sciences Department, September 1991.

[LAH88] Laha, S; Patel, J.K., IYER, R.K. "Accurate Low_Cost Methods for Performance Evaluation of Cache Memory Systems", IEEE Transactions on Computers, Vol 37, No 11, pp. 1325-1336, Nov. 1988.

[LAU93] Lauterbach,G. "Accelerating Architectural Simulation by Parallel Execution of Trace Samples ", Sun Microsystems, SMLI TR-93-2, December 1993.

[LEE84] Lee, J.K.F.; Smith,A.J. "Branch Prediction Strategies and Branch Target Buffer Design", IEEE Computer, January 1984.

[LIU93] Liu, L and peir, J-K. "Cache Sampling by Sets" IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.1, No. 2, pp 98-105, June 1993.

[MAR93] Martonosi, M; Gupta, A; Anderson, T. "Effectiveness of Trace Sampling for Performance Debugging Tools", In Proc ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, June 1993.

[PLE94] Pleszkun, A. R. "Techniques for Compressing Program Address Traces" Proc. of the 24 th Annual Internationa Symposium on Microarchitecture, San Jose, CA, pp. 32-39, 1994

[SHI94] Shipnes, J. and Phillip, M. "The PowerPC Performance Modeling" Communications of the ACM, Vol. 37, No 6, pp 47-63, June 1994.

[TOM67] Tomasulo, R.M. "An Efficient Algorithm for Exploiting Multiple Arithmetics Units", IBM Journal, January 1967.