Avaliação da transação de endereço de memória virtual para física usando TLB de 4KB e 4MB

Paulo H. Ribeiro[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (Unicamp)
Campinas – SP – Brazil

paulohsilvar@gmail.com

Abstract. *This reporting documentation is to describe the project2 of MO601 discipline of Computing Institute of UNICAMP. The study is the evaluation of memory usage when using TLB 4KB or 4MB TLB. 10 SPEC2006 benchmarks were chosen, the criteria evaluated were total access to main memory for instructions and data, total misses in the instruction and data TLB and total access tables pages for instruction and data. A toy_benchmark was also created for environmental assessment.*

Resumo. *Essa documentação de reporte vem para descrever o projeto2 da disciplina MO601 do Instituto de Computação da UNICAMP. O trabalho consiste na avaliação do uso de memória quando usamos TLB de 4KB ou TLB de 4MB. Foram escolhidos 10 benchmarks do SPEC2006, os critérios avaliados foram o total de acesso a memória principal por instrução e por dados, o total de misses na TLB de instrução e de dados e o total de acesso as tabelas de páginas para instrução e dados. Um toy_benchmark também foi criado para avaliação do ambiente.*

1. Introdução

Um trabalho comum para os processadores hoje é a conversão de endereço virtual para endereço físico, a razão de se usar endereço virtual é buscar alcançar a sensação do sistema computacional possuir mais memória, oferecer maior proteção aos programas, layout de memória, entre outros. A quantidade de memória dos sistemas sempre foi limitada, oferecer o sentimento de ter mais memória permite que se use mais programas, isso se dá pelo fato de eles receberem endereços virtuais para suas execuções e não os endereços físicos em que eles realmente estão, logo com o uso de memória virtual os programas não tem acesso a real posição de memória que outros programas estão, evitando assim programas mal intencionados de agirem sobre outros. No layout de memória pode-se carregar uma biblioteca e cada programa que vai usá-la recebe um endereço virtual dela, assim ela só precisa ser carregada uma vez e vários programas conseguem acessá-la.

É nítido perceber a importância de usarmos endereços virtuais, mesmo acrescentando assim um trabalho a mais ao sistema computacional, o de realizar a conversão dos endereços para efetuar suas execuções. A nível de processador, a TLB (translation lookaside buffer) é uma cache responsável por realizar essa tarefa, armazenando a conversão dos dois endereços. A necessidade dessa cache se dá porque as tabelas de páginas encontram-se na memória principal, e teríamos um custo altíssimo

para realizar a conversão de todos os endereços acessando a memória principal.

Nesse projeto vamos avaliar como se comporta o sistema quando usamos uma TLB de 4KB contra uma TLB de 4MB, os critérios de avaliação definidos foram a quantidade total de acessos a memória por instrução e por dados, a quantidade total de TLB misses por instrução e por dados e a quantidade de acessos as tabelas de páginas por instrução e por dados.

2. Ferramentas utilizadas

As ferramentas utilizadas para executar esse projeto foram o PIN da intel e o SPEC2006, buscando alcançar um melhor desempenho em tempo de execução foi utilizado a técnica de simpoints para rodar os benchmarks. Os pinballs utilizados foram os do cpu2006-pinpoints-w0-d1B-m1. Assim todos os benchmarks do SPEC2006 foram executados para essa avaliação, para apresentação dos resultados apenas os 10 benchmarks que apresentaram os maiores números de acesso a memória de dados foram considerados.

Usando essa métrica, os benchmarks selecionados foram os seguintes: mcf, milc, libquantum, hmmer, gcc, leslie3d, lbm, soplex, omnetpp, zeusmp. Para apresentação de resultados gráficos com relação aos benchmarks que possuem mais de uma entrada, apenas a entrada com maior resultado foi utilizado.

3. Acesso à memória

O primeiro critério de avaliação foi a quantidade de acessos a memória por instrução e por dados. O processador só acessa a memória principal quando o que ele procura não se encontra em nenhum dos seus níveis de cache, ou seja, quando todos os acessos a uma instrução ou dado obtiveram misses até o seu último nível, o próximo passo é acessar a memória. Os níveis de cache no processador conforme modelados pelo pintool desenvolvida, apresenta apenas o primeiro nível separado para instrução e dados, os demais níveis são unificados, desta forma, o nível de cache que nos apresenta que haverá um acesso a memória é um só tanto para dados quanto para instrução, é preciso reconhecer a origem desse acesso para poder defini-lo. Na Figura 1 a seguir temos o resultado obtido para instrução e na Figura 2 para dados.

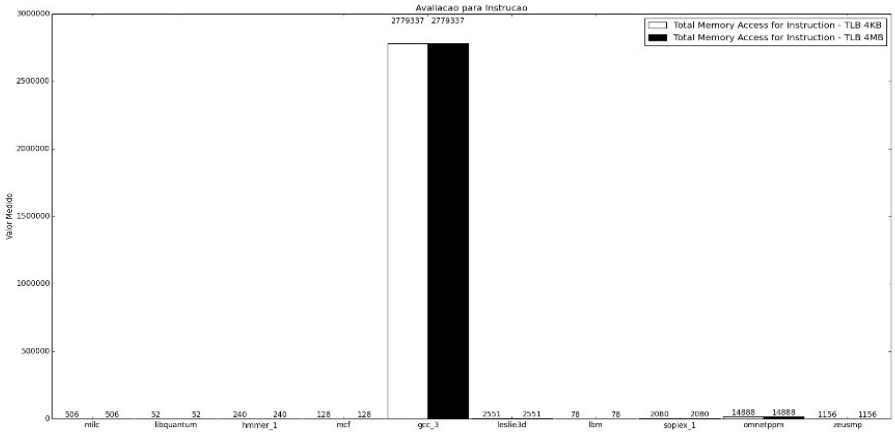

Figura 1. Total de acessos a memória por Instrução

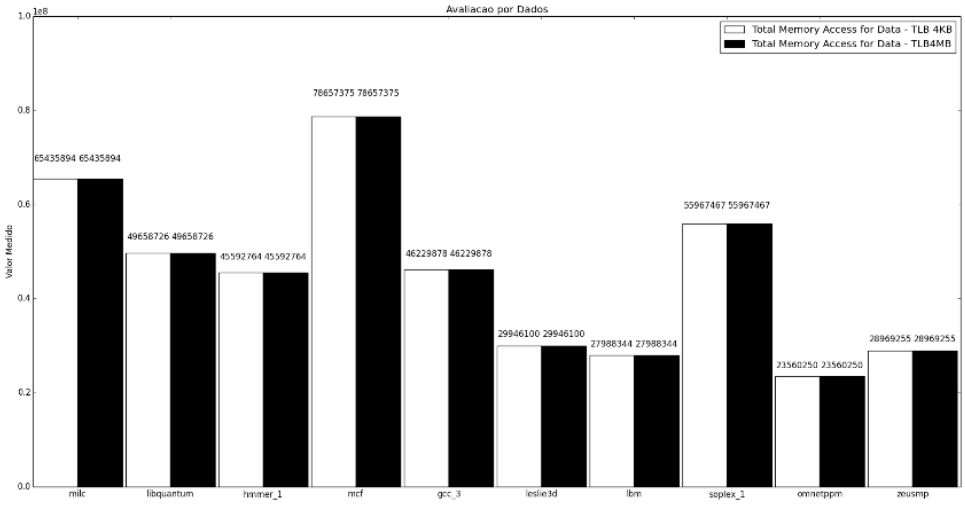

Figura 2. Total de acessos a memória por Dados

Como podemos ver pela Figura 1 e Figura 2, o acesso a memória tanto de dados como de instrução não teve relação com o tamanho da TLB que utilizamos, justificando a mesma quantidade de acessos, isso ocorre porque na pintool não há relação entre a TLB e os níveis de cache

4. Misses na TLB

O segundo critério avaliado foi a quantidade de misses na TLB, como visto anteriormente ela é uma cache para armazenar as tabelas de páginas que são responsáveis por fazer a tradução de endereço virtual para físico.

Quando o processador recebe uma operação, ela possui um endereço virtual, ocorre um acesso a TLB buscando obter o seu endereço físico sem precisar acessar as tabelas de páginas, pois elas se encontram na memória principal. Ao realizar o acesso a TLB, se essa página virtual não estiver lá ocorre um miss nessa cache. Na Figura 3 e na Figura 4 a seguir, temos os resultados para instrução e dados respectivamente.

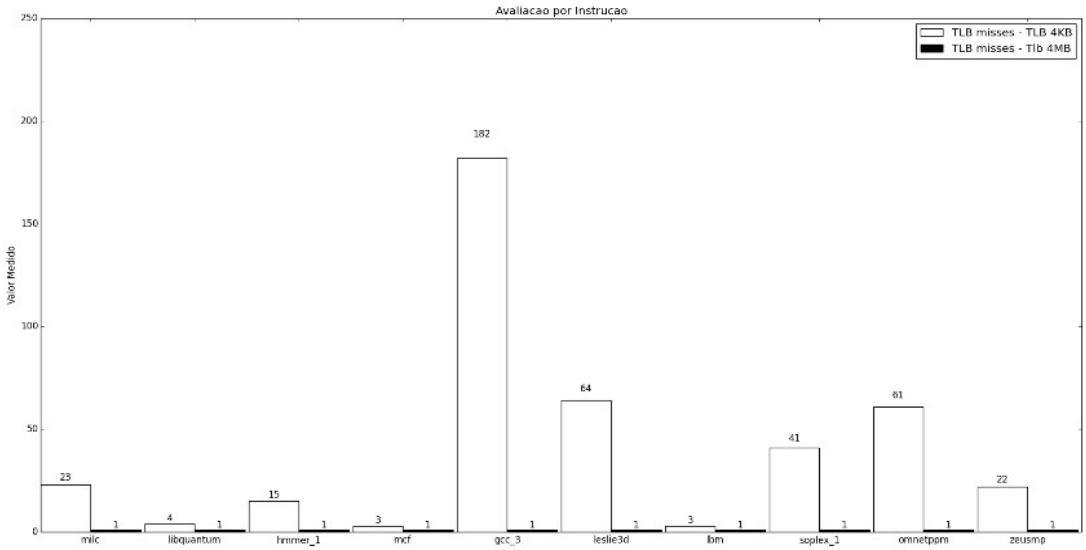

Figura 3. Total de misses na TLB para instrução

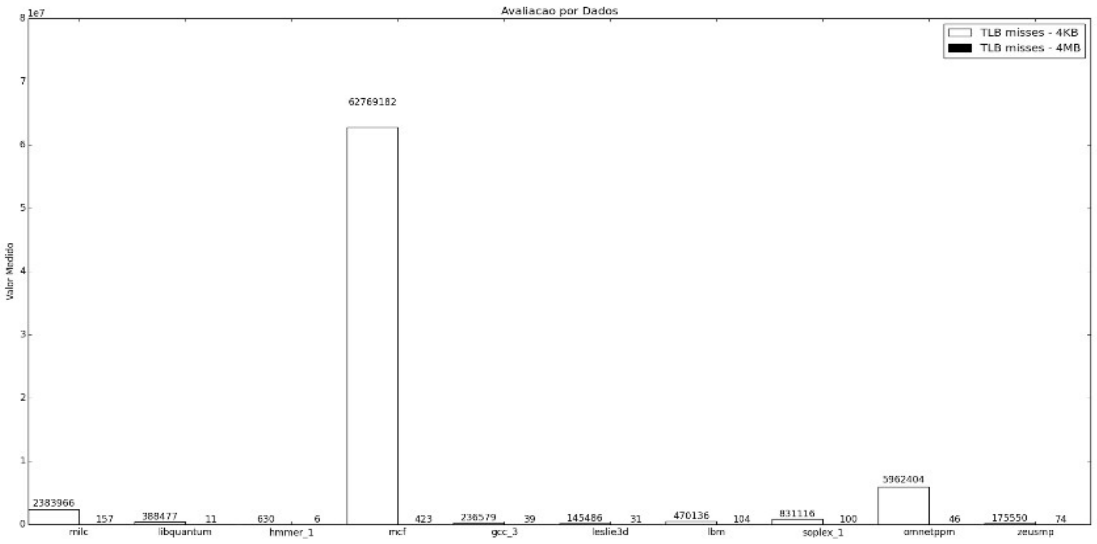

Figura 4. Total de misses na TLB para dados

Analisando os resultados contidos nos gráficos, percebemos o quão relevante é o tamanho da TLB para a quantidade de misses por ela produzida. Para a Figura 3, a menor diferença temos três vezes mais quantidade de misses, ocorrido no mcf. Quando observamos a Figura 4, percebemos a menor diferença sendo de cento e cinco vezes, ocorrida no hmmer_1. Essa diferença ocorre pois ao se carregar a TLB, o tamanho de páginas trazidas serão de 4KB ou 4MB, usando a segunda opção trazemos muito mais informação do que para primeira, assim quando novos endereços são acessados a chance de encontrá-los na TLB é muito maior.

5. Total de tabelas de páginas acessadas

O último critério avaliado foi a quantidade de tabelas de páginas acessadas. Essas tabelas se encontram na memória principal e são acessadas quando não se consegue realizar a conversão de endereço virtual para físico pela TLB. Foi considerado que as tabelas de páginas possuíam 3 níveis. Nas Figura 5 e Figura 6 temos os resultados para instrução e dados respectivamente.

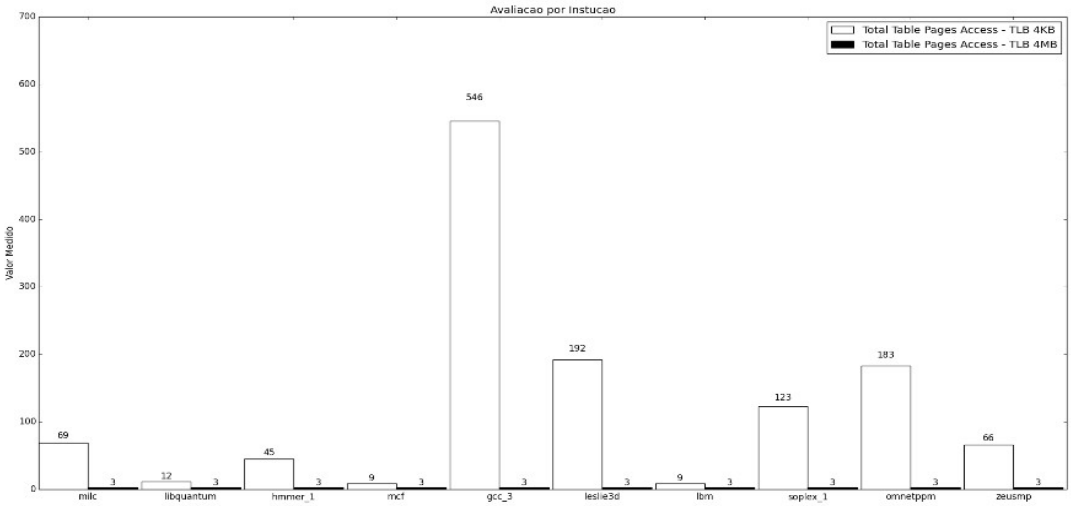

Figura 5. Total de tabelas de páginas acessadas para instrução

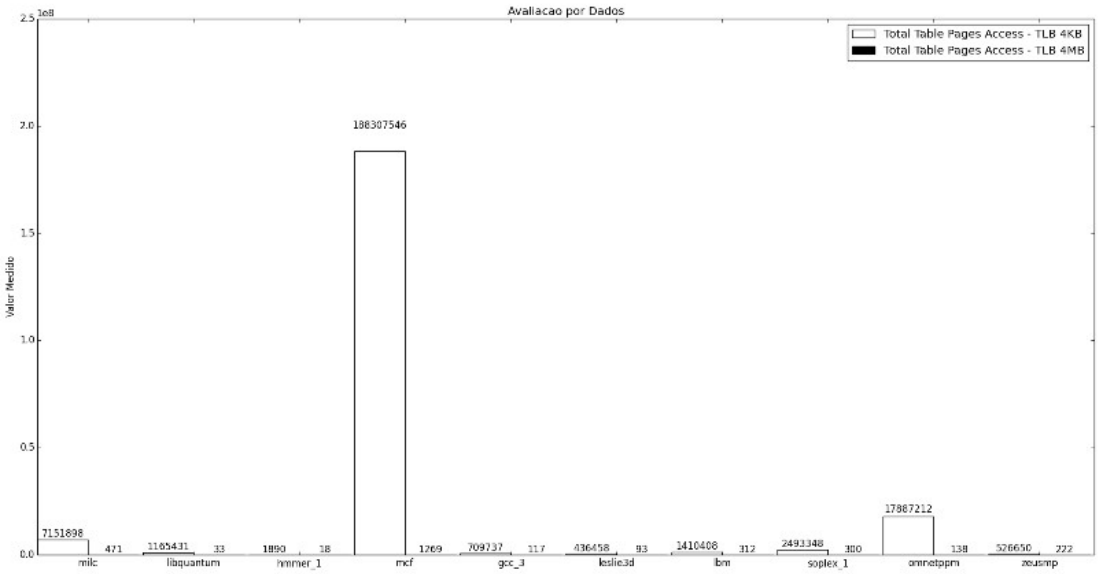

Figura 6. Total de tabela de páginas acessadas para dados

Analisando os resultados dos gráficos, podemos perceber que a quantidade de tabelas de páginas acessadas quando temos TLB de 4KB é tão superior a de 4MB como a quantidade de misses na TLB. Essa relação existe pelo fato de que as tabelas de páginas são acessadas quando o miss na TLB ocorre, como temos 3 níveis de tabelas de páginas, cada vez que é preciso buscá-las na memória, temos três acessos a essas tabelas.

6. Toy benchmark

Para validação do ambiente desenvolvido de avaliação, foi criado um toy_benchmark, conforme apresentado na Figura 7 a seguir.

```
#include<iostream>
using namespace std;

int main()
{

    int size = 100000;
    int next_pos = 4500;

    int pos = 0;
    int vector[100000] = {};


    for(int i=0; i<100; i++){
        vector[pos] = i;
        pos = pos + next_pos;
        if(pos >= size) {
            pos = pos - size;
        }
    }
    return 0;
}
```

Figura 7. Toy_benchmark

Ele tem um array com cem mil posições, seu loop possui cem iterações que

consiste de armazenar um valor na posição anterior mais quatro mil e quinhentos, onde a posição inicial é zero, quando a posição a ser acessada é maior que o tamanho do array, ele recomeça das posições iniciais. Temos a Figura 8 a seguir que apresenta seus resultados no mesmo molde do .csv dos benchmarks.

```
toy_benchmark4kb 1928 91 273 115504 247 741
toy_benchmark4mb 1936  3   9 115597   5  15
```

Figura 8. Toy_benchmark resultados

Como podemos perceber o número de TLB misses para 4KB foi próximo do número de iterações no array, isso porque acessava sempre uma posição 4K de diferença da atual. Para a TLB de 4MB já não houve essa quantidade pelo seu tamanho ser bem maior.

7. Referências

Pinplay,

http://snipersim.org/w/Pinballs

TLB widipedia,

https://en.wikipedia.org/wiki/Translation_lookaside_buffer