Universidade Federal de Santa Catarina Centro Tecnológico - CTC
Departamento de Informática e Estatística - INE

Relatório atividade 5

Organização de Computadores I

Prof.
Marcelo Daniel Berejuck

Alunos
Augusto de Hollanda Vieira Guerner (22102192) e
Fabricio Duarte Júnior (22100615)

Relatório atividade 5

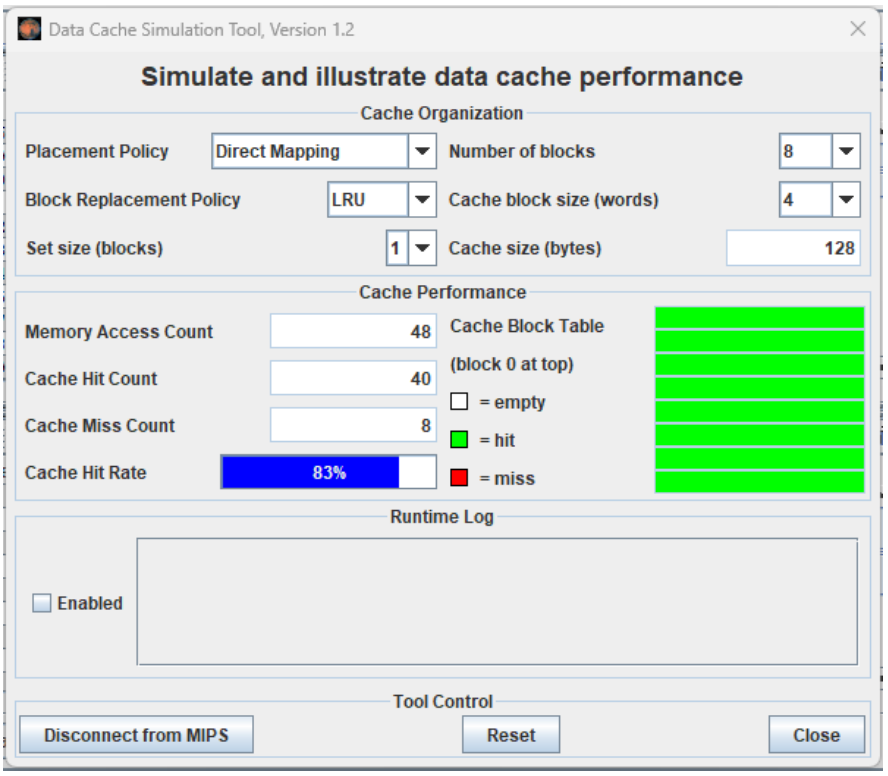
Introdução

A presente atividade tem por objetivo estudar blocking cache, comparando o desempenho de dois códigos que essencialmente fazem a mesma coisa (a soma da matriz A com a transposta de B), mas de modos diferentes. O primeiro faz o percurso na matriz de forma padrão, isto é, percorrendo linha por linha e coluna por coluna. O segundo diferentemente percorre bloco por bloco, visando otimizar o uso da cache. Mas antes, vale destacar que o algoritmo apresentado de blocking não leva em consideração tamanhos de matrizes que não são múltiplos do tamanho do blocking, assim, para certos valores, o resultado pode eventualmente ser inconsistente.

Exercício 1)

Neste primeiro exercício, foi solicitada a implementação de um algoritmo que fizesse a soma de uma matriz A com a transposta de B usando uma lógica padrão, ou seja, percorrendo linha por linha da matriz A e coluna por coluna da matriz B. Depois disso, foi avaliado o desempenho do programa usando a simulação de cache. Antes de prosseguir com os resultados e suas análises, é interessante notar que o algoritmo escrito não precisa definir valores para os elementos das matrizes A e B, já que o que está sendo avaliado é o acesso à cache dentro dos *loops* da soma.

Imagem 1 - Simulação da cache exercício 1 com tamanho da matriz igual a 4.



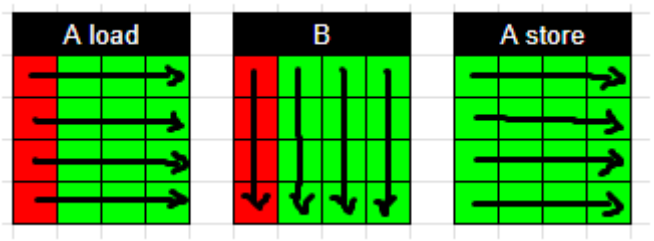
Fonte: Acervo dos autores.

Como se pode observar na **imagem 1**, a taxa de acertos quando a cache tem 8 blocos e 4 words em cada, e as matrizes têm tamanho 4, é de 83%, isto é, de 48 acessos à memória 40 são acertos e 8 são miss. O comportamento que resultou nesta taxa de acertos é o seguinte:

- No load da matriz A, ocorre 1 miss no primeiro elemento de cada linha, seguido de 3 hits, ou seja, 1 miss a cada elemento que inicia um novo bloco da cache;
- No load da matriz B, ocorrem 4 misses seguidos, um para cada linha, seguido apenas de hits. Da mesma forma, é 1 miss a cada elemento que inicia um novo bloco da cache;
- Após os misses, como a cache tem um número de blocos que comporta as duas matrizes, elas já estão completamente na cache. Em razão disso, não ocorre miss algum no store da matriz A.

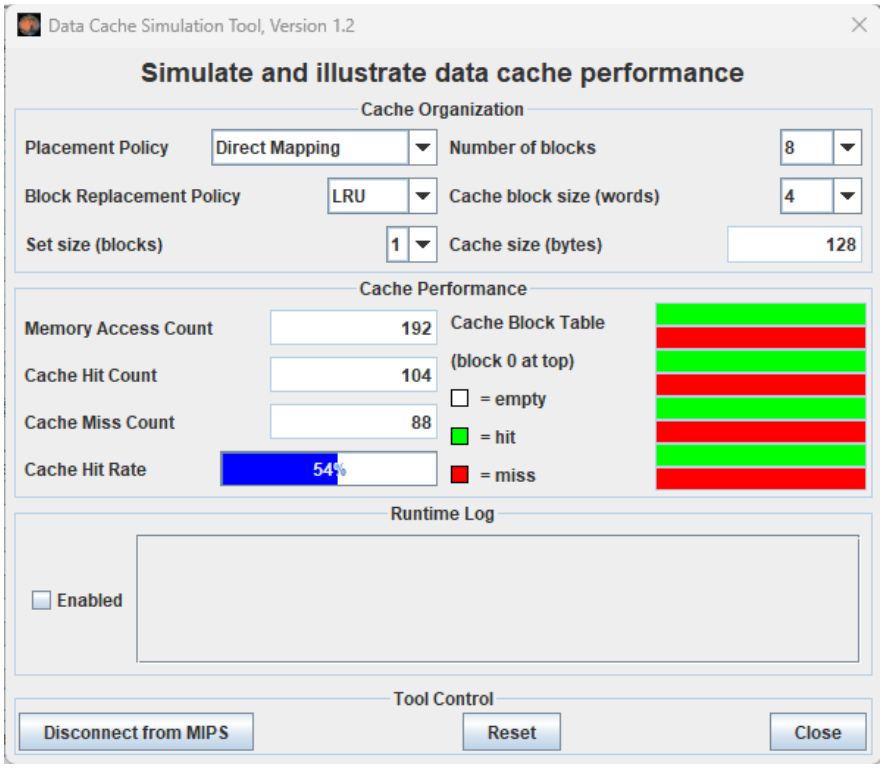
Abaixo está uma imagem em que é possível visualizar as posições em que ocorrem misses e hits em cada matriz:

Imagem 3 - Ilustração dos MISSes e dos HITs da imagem 1.



Fonte: Acervo dos autores.

Imagem 2 - Simulação da cache exercício 1 com tamanho da matriz igual a 8.



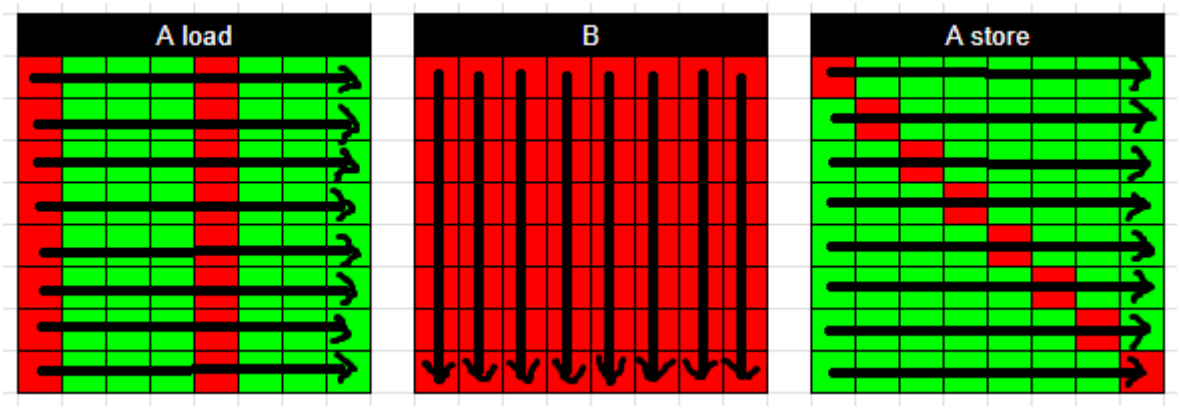
Fonte: Acervo dos autores.

Na segunda situação, em que as matrizes têm tamanho 8, a taxa de acertos diminui consideravelmente. Isto ocorre porque a cache tem um tamanho que não comporta as duas matrizes e que resulta nas mesmas posições das matrizes A e B sendo mapeadas para os mesmos blocos, fazendo com que eles sejam sobrescritos. O comportamento nessa situação é o seguinte:

- O load da matriz A continua com 3 hits a cada 1 miss;
- O acesso à matriz B sempre falha, pois, toda vez em que um elemento vai ser acessado, o bloco correspondente já foi sobrescrito pelo load da matriz A;
- O store da matriz A falha somente na diagonal principal, pois o load da B acabou de sobrescrever o bloco equivalente.

Na imagem abaixo podem ser visualizados os elementos em que ocorreram as falhas:

Imagem 4 - Ilustração dos MISSes e dos HITs da imagem 2.



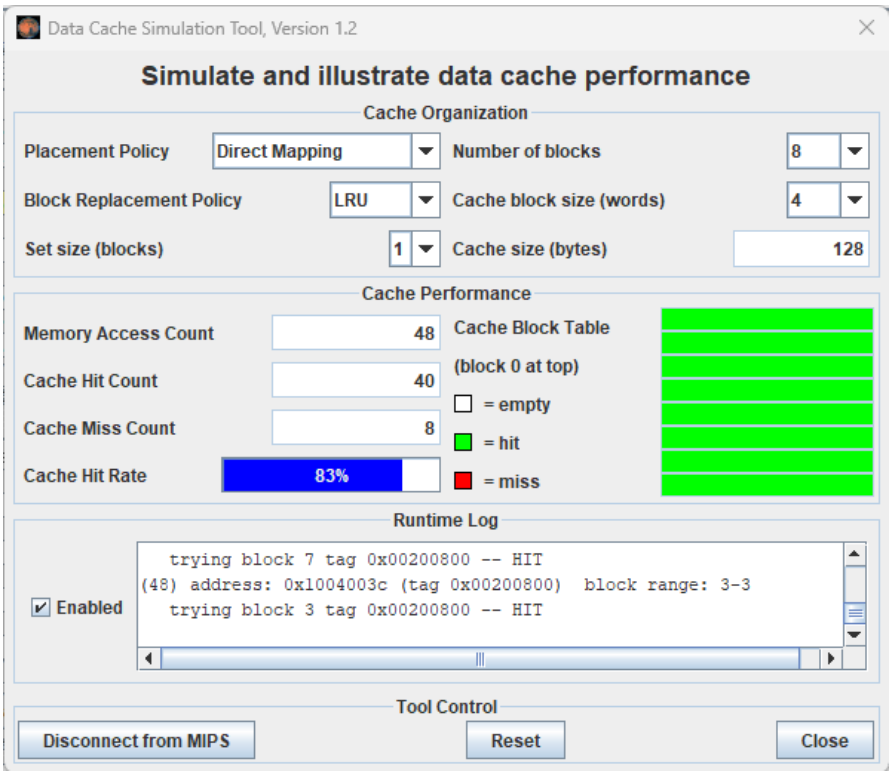
Fonte: Acervo dos autores.

Assim, o resultado é de 104 hits e 88 misses.

Exercício 2)

Neste segundo exercício, foi solicitada a implementação de um algoritmo que fizesse a soma de uma matriz A com a transposta de B usando uma lógica voltada à otimização do uso da cache. Assim, tanto a matriz B quanto a A eram divididas em blocos e, a partir disso, aplicava-se o algoritmo anterior de soma de matriz em cada um desses blocos, um por vez. Depois disso, foi avaliado o desempenho do programa usando a simulação de cache. No mais, neste exercício, tal qual o 1, também não era necessário definir valores para os elementos das matrizes.

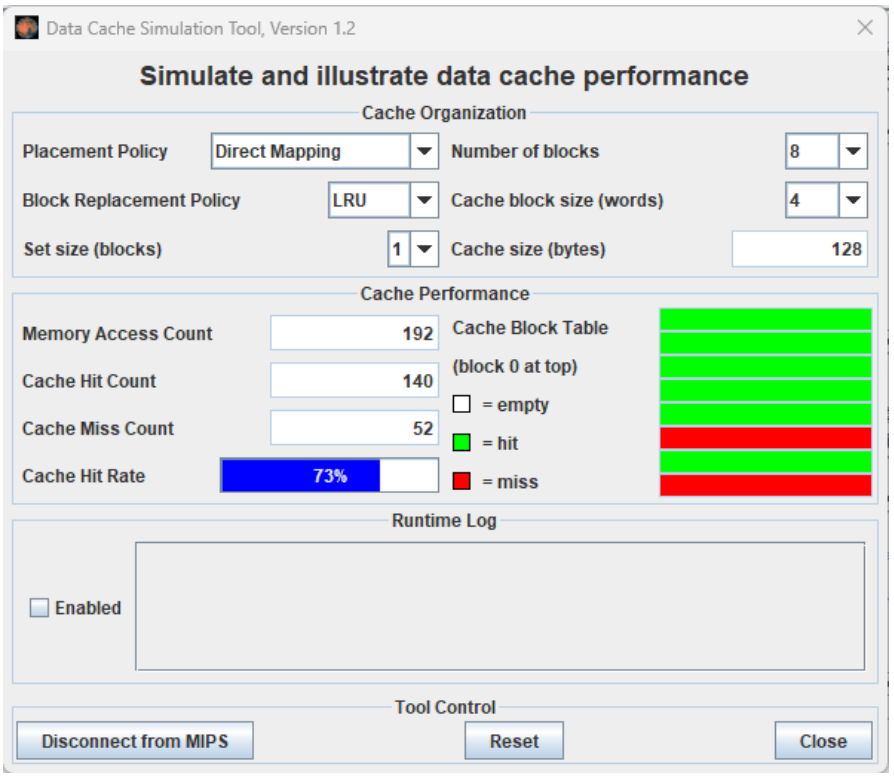
Imagem 5 - Simulação da cache exercício 2 com tamanho da matriz igual a 4 e tamanho do bloco igual a 4.



Fonte: Acervo dos autores.

Ao definir o tamanho das matrizes igual ao tamanho do bloco, neste caso, 4, não há diferença entre utilizar ou não utilizar cache blocking no quesito de acertos. Isto ocorre porque, como o tamanho da matriz é igual ao tamanho do bloco, o algoritmo funciona da mesma forma que o código do primeiro exercício.

Imagem 6 - Simulação da cache exercício 2 com tamanho da matriz igual a 8 e tamanho do bloco igual a 4.



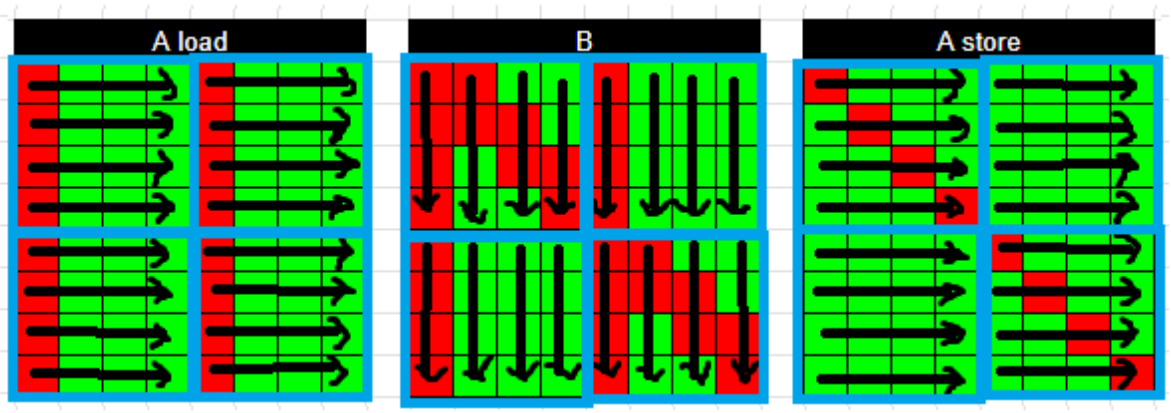
Fonte: Acervo dos autores.

Na situação com tamanho de matrizes igual a 8 e tamanho de bloco da cache igual a 4, ao comparar com o resultado do último exemplo do exercício anterior, é possível perceber que a técnica de cache blocking resultou em uma melhoria significativa da taxa de acertos. O novo comportamento é o seguinte:

- O load da matriz A continua com 1 miss seguido de 3 hits;
- No load da matriz B:
    - Falha nos elementos que iniciam um novo bloco da cache (cada 4 elementos);
    - Falha em todos os elementos da diagonal, pois o bloco foi sobrescrito pelo load da A;
    - Falha em todos os elementos que sucedem a diagonal, pois, mesmo tendo carregado o bloco após falhar na diagonal, ele é sobrescrito pelo store da A.
- O store da matriz A continua falhando na diagonal, pois o bloco foi sobrescrito pelo acesso à B.

  Os elementos em que ocorreram falhas podem ser visualizados abaixo:

Imagem 7 - Ilustração dos MISSes e dos HITs da imagem 6.



Fonte: Acervo dos autores.

Assim, apesar de a matriz B ainda falhar bem mais que a matriz A, foi possível aproveitar a localidade espacial da memória e reduzir o número de falhas em relação ao exercício 1.

Conclusão

A partir do supracitado, percebe-se que a atividade 5, além de necessitar conhecimento teórico sobre o conteúdo blocking cache para o correto entendimento dos códigos em C, também exigiu um maior conhecimento sobre as instruções do MIPS, como a alocação de memória em tempo de execução. Assim, mais uma vez os alunos foram tirados de sua zona de conforto, tal qual nas atividades anteriores, de modo que precisaram aprender assuntos novos.