
# Trabalho Prático:

# Simulação e Análise de Memória Cache.

**DISCIPLINA:** ARQUITETURA DE COMPUTADORES.

**ALUNO:** VICTOR DALLAGNOL BENTO, matrícula número 201520835.

**PROFESSOR:** MATEUS BECK RUTZIG.

## INTRODUÇÃO

Neste trabalho foi emulado uma memória cache de instruções e uma memoria cache de dados através do terminal, utilizando o simulador *SimpleScalar Sim-Cache.* Foram utilizados dois benchmarks - FFT1 (Transformada de Fourier) e MM (Multiplicação de Matrizes). O numero de instruções utilizado foi de 40milhões, o numero de conjuntos, tamanho do bloco, tamanho da cache e associatividade foram mudados de acordo com os requisitos exigidos pelo professor.

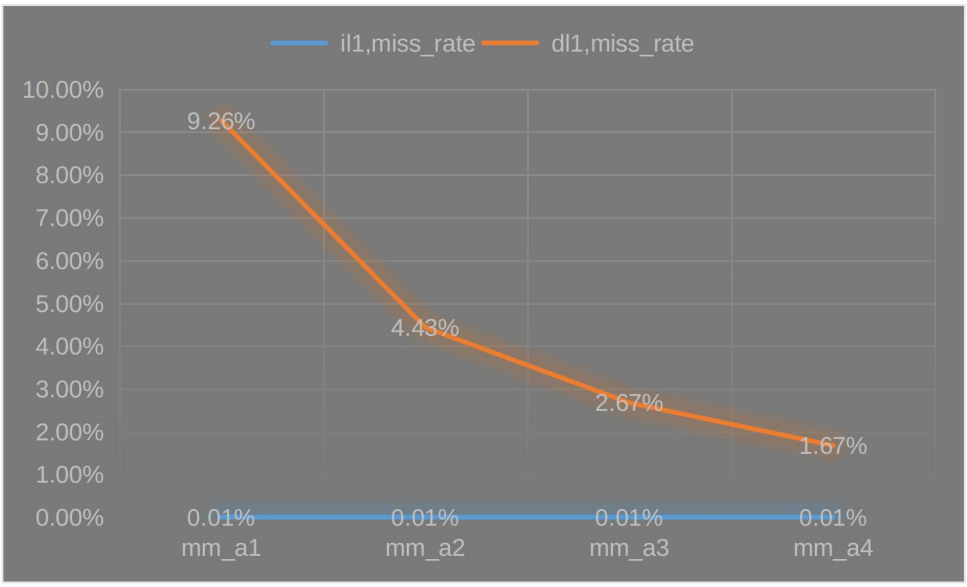
Após a execução e a obtenção dos dados, uma analise detalhada dos resultados será apresentada.

**A. Influência do tipo de mapeamento (direto, associativo por conjunto e totalmente associativo):**

O tamanho da Cache, tanto de dados como de instruções para esta alternativa foi fixada em 2 Kbytes cada.

→ Multiplicação de Matrizes:

| *alternativa* | *Nº Conjunto* | *Tamanho Bloco* | *Associatividade* | *il1,miss_rate* | *dl1,miss_rate* |
|---|---|---|---|---|---|
| mm_a1 | 64 | 32 | 1 | 0,0001 | 0,0926 |
| mm_a2 | 16 | 32 | 4 | 0,0001 | 0,0443 |
| mm_a3 | 8 | 32 | 8 | 0,0001 | 0,0267 |
| mm_a4 | 1 | 32 | 64 | 0,0001 | 0,0167 |




Cache de Instruções: Independente do mapeamento ela manteve uma taxa de *misses* constantes de 0,01%, isto se deu pelo fato do benchmark executar muitas instruções sequenciais, percorrendo linhas de determinadas colunas, por possuir 8 palavras por bloco(localidade espacial), varias instruções de endereços contíguos são trazidos da memória e executados sequencialmente. O código provavelmente possui muitos loops sequenciais, e por ter política LRU, e ter apenas acessos de leitura, seu desempenho foi excelente.

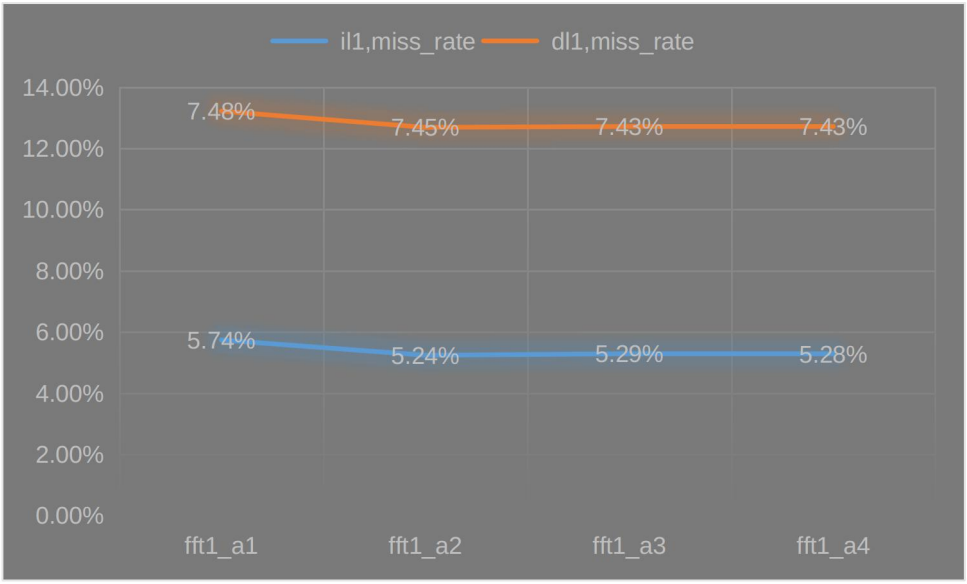
Cache de Dados: Por causa da política de LRU, quando aumentamos a associatividade o numero de *misses* diminui bastante, devido a localidade temporal. Porém, podemos observar que essa diminuição não é constante. Quando diminuamos o numero de conjuntos, por consequência do aumento da associatividade, o numero de misses aumenta.

A melhor combinação entre Cache de Dados e Instruções seria a **mm_a3**, que tem associatividade 8-way. Com isso o algoritmo de substituição só considera blocos dentro de um conjunto. Apesar de possuir uma taxa de *misses* um pouco maior do que a

completamente associativa (**mm_a4**), seu gasto com comparadores seria de 8x menor.

→ Transformada de Fourier:

| alternativa | Nº Conjunto | Tamanho Bloco | Associatividade | il1,miss_rate | dl1,miss_rate |
|---|---|---|---|---|---|
| fft1_a1 | 64 | 32 | 1 | 0,0574 | 0,0748 |
| fft1_a2 | 16 | 32 | 4 | 0,0524 | 0,0745 |
| fft1_a3 | 8 | 32 | 8 | 0,0529 | 0,0743 |
| fft1_a4 | 1 | 32 | 64 | 0,0528 | 0,0743 |




Cache de Instruções: Em relação a aplicação MM, a Transformada de Fourier (FFT1) possui um maior numero de *misses* (aprox. 6%) por parte das instruções, isso pode ter ocorrido pelo fato do programa usar loops que não são executados sequencialmente de um loop para outro (um for dentro de outro for, por exemplo), o que faz com que alguns endereços que não foram usados recentemente sejam sobre-escritos.

Cache de Dados: Quase não tem melhora na taxa de *misses* independente do numero de conjunto/associatividade, possivelmente pelo fato do programa ocupar toda memória - esta sempre reescrevendo-, e pelo fato de possuir muitos loops, um dentro do outro, o que faz com que as instruções não sejam executadas sequencialmente.

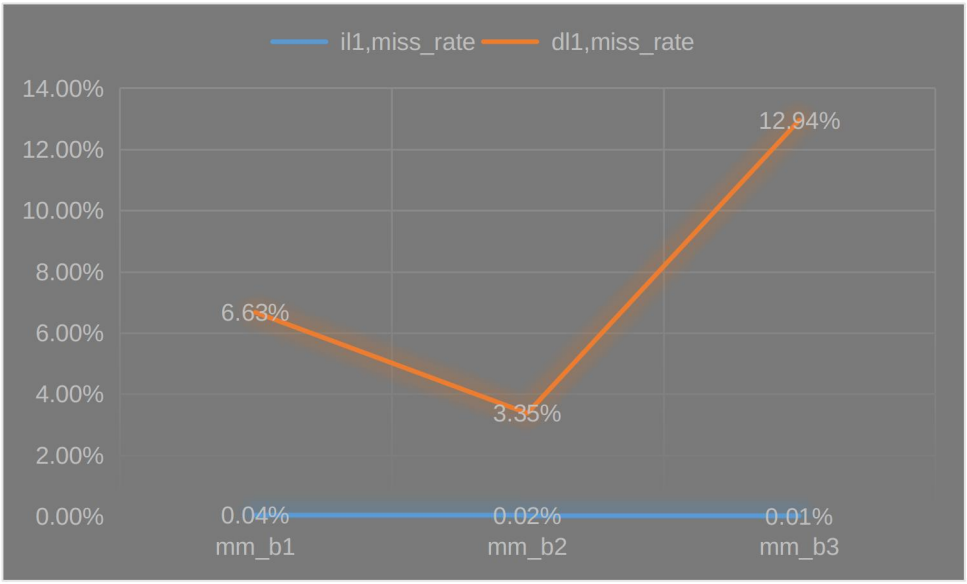
Por possuir localidade espacial de 8 palavras por bloco, quando acontece um *Miss,* 8 palavras são perdidas. A mudança no numero de conjuntos e na associatividade não tem muita significância por causa da forma com que o programa é executado (loops). A melhor cache (de instruções e dados) seria a mapeada diretamente (**fft_a1**) pois não precisaria de um algoritmo de substituição e teria um desperdício menor de hardware.

**B. Investigar a influência da variação do tamanho do bloco no desempenho da cache:**

O tamanho da Cache, tanto de dados como de instruções para esta alternativa foi fixada em 2 Kbytes cada

→ Multiplicação de Matrizes:

| *alternativa* | *Nº Conjunto* | *Tamanho Bloco* | *Associatividade* | *il1,miss_rate* | *dl1,miss_rate* |
|---|---|---|---|---|---|
| *mm_b1* | 64 | 8 | 4 | 0,04% | 6,63% |
| *mm_b2* | 32 | 16 | 4 | 0,02% | 3,35% |
| *mm_b3* | 8 | 64 | 4 | 0,01% | 12,94% |




Cache de Instruções: Apresentou pouca melhora com o aumento do tamanho do bloco, tendo uma variação de 0,03%. Quando aumentamos o tamanho do bloco aumentamos também a localidade espacial, mais palavras contíguas são buscadas da memória, a associatividade é fixa em 4-way (4 blocos em um conjunto).

Cache de Dados: Apresentou melhora de aproximadamente 50% com o aumento do tamanho do bloco e diminuição de conjuntos. A localidade espacial foi explorada, mais dados foram usados sequencialmente pelo loop, por causa da localidade espacial, os blocos buscados já estavam na memória e prontos para o uso. Para a Cache **mm_b3**, o numero de *misses* aumentou drasticamente. Esse aumento, é justificável se pensarmos que toda vez que a multiplicação de matrizes troca de coluna, ela comete um *miss*, e como muitas palavras são buscadas por conta da localidade espacial, temos poucos blocos/conjuntos, o que dificulta a localização do dado.

A Combinação entre as Caches com o melhor resultado é a **mm_b2**, tanto para a cache de dados quanto para a de instruções, visto que a variação da cache de instruções é pequena e a melhor cache de dados é a **mm_b2**.

→ Transformada de Fourier:

| *alternativa* | *Nº Conjunto* | *Tamanho Bloco* | *Associatividade* | *il1,miss_rate* | *dl1,miss_rate* |
|---|---|---|---|---|---|
| *fft1_b1* | 64 | 8 | 4 | 16,73% | 28,43% |
| *fft1_b2* | 32 | 16 | 4 | 9,22% | 14,54% |
| *fft1_b3* | 8 | 64 | 4 | 3,01% | 3,86% |


Cache de Instruções: Com o aumento do bloco e a diminuição do conjunto, a cache teve uma diminuição no numero de *misses*. Como mencionado na alternativa A, o programa da transformada de Fourier deve executar muitos loops fora de sequencia, e como a localidade espacial esta fortemente explorada, as instruções são encontradas facilmente.

Cache de Dados: Pelo mesmo princípio da cache de instruções a cache de dados também teve uma melhora significativa na taxa de *misses*. A localidade espacial ajuda na procura de dados, facilitando seu acesso. Com o aumento da L.E mais dados são trazidos para o mesmo bloco da cache.

A cache mais eficiente é a **fft1_b3**, para dados e instruções. Apesar de ser uma cache com poucas linhas(8), e correndo o risco de perder muitas palavras ao cometer um *miss* ou de ficar difícil de achar o dado, o programa da transformada de fourier explora bem a localidade espacial dos dados.

Comparando ambas as memórias das alternativas A e B, as melhores caches seriam:

- MM: Para a cache de instruções qualquer uma da **alternativa A** poderia ser escolhida, todas tem a mesma taxa de *miss* independente do seu mapeamento. Para a cache de dados a melhor opção seria a **mm_b2**, da alternativa B. Essa cache de dados se beneficia da localidade espacial explorada.

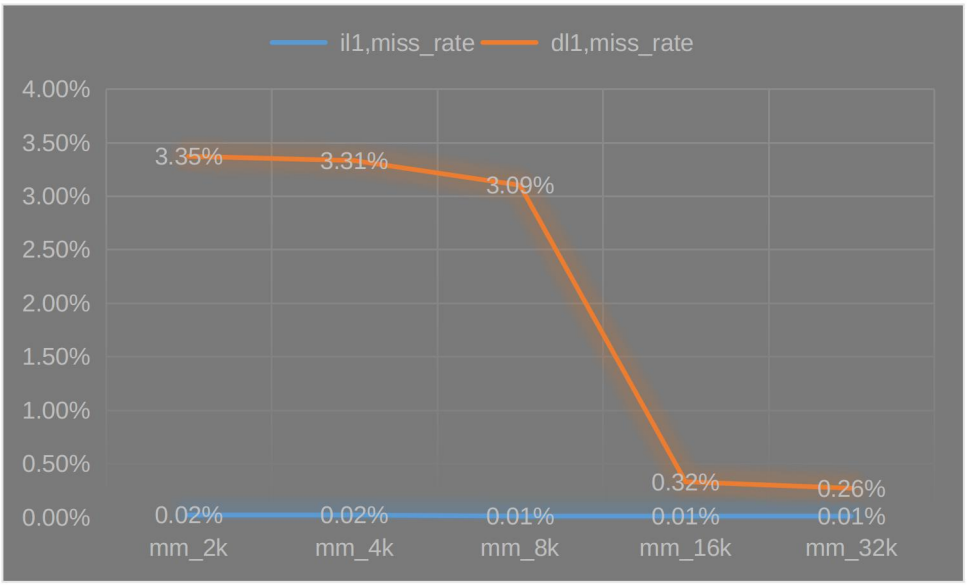
- FFT1: A melhor Cache de instruções e de dados é a mesma, **fft1_b3**, por que ambas as caches são beneficiadas com o localidade espacial. Por possuir loops dentro

de loops, o programa pega todos os dados/instruções de um loop em um mesmo bloco. È Claro que os misses existentes se dão pelo fato da saída de um loop interno e entrada no loop externo.

## C. Investigar a influência do tamanho total da cache:

→ Multiplicação de Matrizes:

| *alternativa* | *Tamanho Total* | *Nº Conjunto* | *Tamanho Bloco* | *Associatividade* | *il1,miss_rate* | *dl1,miss_rate* |
|---|---|---|---|---|---|---|
| *mm_2k* | 2k | 32 | 16 | 4 | 0,02% | 3,35% |
| *mm_4k* | 4k | 64 | 16 | 4 | 0,02% | 3,31% |
| *mm_8k* | 8k | 128 | 16 | 4 | 0,01% | 3,09% |
| *mm_16k* | 16k | 256 | 16 | 4 | 0,01% | 0,32% |
| *mm_32k* | 32k | 512 | 16 | 4 | 0,01% | 0,26% |

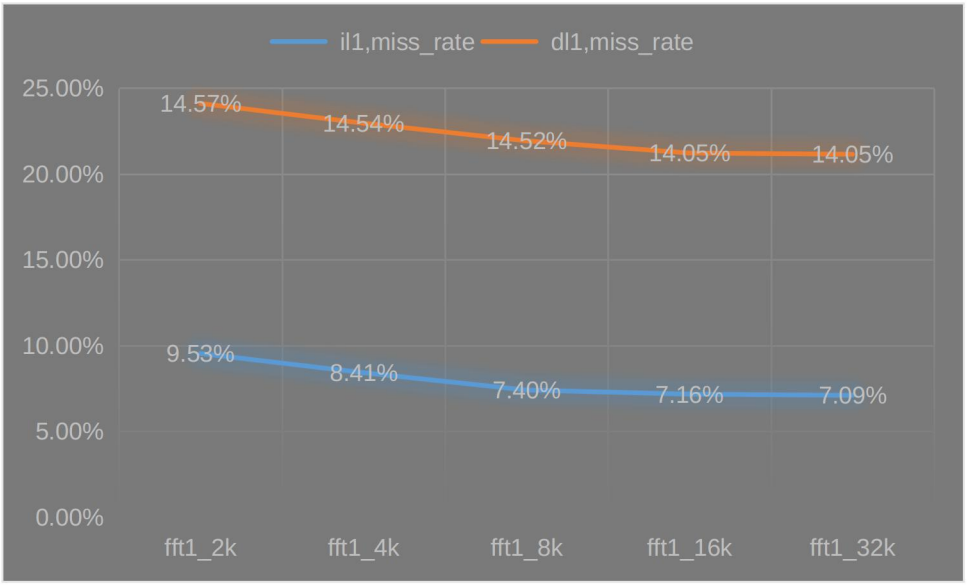



A melhor opção de cache de instruções é a de 2k (**mm_2k**), que em relação as outras não tem tanta diferença na taxa de *misses*, e por ter um tamanho menor, tem um menor desperdício.

A melhor cache de dados é a **mm_16k**. Em comparação com a de 32k ela possui uma taxa de misses um pouco mais elevada, mas levando em conta que tem a metade do tamanho(menos desperdício), com praticamente a mesma eficiência, ela se torna a melhor opção. A cache de 16k já explora ao máximo a localidade espacial, aumentar seu tamanho seria desperdício.

→ Transformada de Fourier:

| alternativa | Tamanho Total | Nº Conjunto | Tamanho Bloco | Associatividade | il1,miss_rate | dl1,miss_rate |
|---|---|---|---|---|---|---|
| *fft1_2k* | 2k | 32 | 16 | 4 | 9,53% | 14,57% |
| *fft1_4k* | 4k | 64 | 16 | 4 | 8,41% | 14,54% |
| *fft1_8k* | 8k | 128 | 16 | 4 | 7,40% | 14,52% |
| *fft1_16k* | 16k | 256 | 16 | 4 | 7,16% | 14,05% |
| *fft1_32k* | 32k | 512 | 16 | 4 | 7,09% | 14,05% |




Para a Transformada de Fourier, a melhor cache de instruções é a de 8k (**fft1_8k**). Tem pouca diferença entre as de 16k e 32k, além de ser menor. Possui diferença de 1% e 2% das memorias de 4k e 2k respectivamente.

A melhor cache de dados é a **fft1_2k**. Ela é a menor dentre as caches (menor desperdício), e possui praticamente a mesma taxa de misses da de 32k, perdendo por 0.5%.