Estudo da Hierarquia de Memória utilizando o simulador Amnesia

1st Arthur Bernardo Coelho Moura
PUC Minas
Belo Horizonte, Brasil
arthurbernardo2334@gmail.com

2nd Gabriel Costa Pinto
PUC Minas
Belo Horizonte, Brasil
gabrielcostap112@gmail.com

3rd Lucas Baesse
PUC Minas
Belo Horizonte, Brasil
lucas.baesse@sga.pucminas.br

***Resumo*—Com o avanço das tecnologias ao longo das décadas, a busca por aperfeiçoamentos no desempenho dos sistemas computacionais no meio científico, tornou-se algo almejado, tanto pela comunidade de pesquisadores e cientistas da computação, quanto por entusiastas de tecnologias. Os sistemas evoluíram tanto em processamento, como em complexidade das arquiteturas implementadas. Nos sistemas computacionais é essencial o uso de memórias, visto que elas são responsáveis pelo armazenamento de dados que serão processados. Considerando um modelo de excelência, as memórias deveriam ter uma capacidade de armazenamento ilimitado, o acesso de dados imediato e o custo por bit extremamente baixo. Entretanto, nos sistemas reais as memórias não apresentam essa natureza. Armazenamento, velocidade e custo por bit são fatores que crescem proporcionalmente e utiliza-se, para balancear esses fatores e melhorar o desempenho dos sistemas computacionais, a hierarquia de memória.**

***Index Terms*—Hierarquia de memória, Amnesia**

I. Introdução

No universo da Computação, compreender a fundo o funcionamento e organização de computadores, é fundamental para dominar habilidades de manipulação e desenvolvimento dessas ferramentas tão importantes na atualidade e, como consequência, contribuir para o surgimento de novas tecnologias.

Um dos componentes mais importantes de um computador é a Memória e sua hierarquia, este é o tema abordado neste artigo. A memória é o dispositivo que possibilita o armazenamento de informações processadas pelo computador, ela pode ser particionada em uma hierarquia em que cada nível contém definições de arquiteturas diferentes.

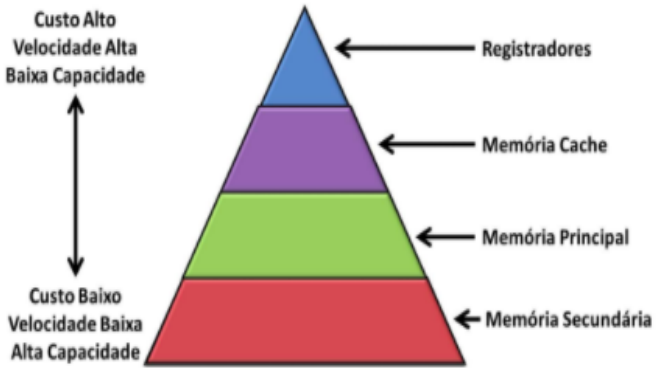

Figura 1. [6] Hierarquia de memória

Como representado na figura 1, os tipos de memória contidos em um computador podem ser representados em uma pirâmide. Na medida em que exploramos os níveis mais altos da pirâmide, cujos representam os níveis da memória, esses níveis se aproximam cada vez mais da CPU, possuem mais velocidade, possuem menos espaço de armazenamento e são financeiramente mais caras. Além disso, para que as memórias da hierarquia tenham um bom desempenho, é preciso explorar os princípios de localidade em sua arquitetura. Esses princípios são:

- Localidade temporal: se uma informação é acessada, ela tende a ser acessada novamente em curto ou médio prazo.
- Localidade espacial: se uma informação é acessada, as informações próximas a esta também tendem a ser acessadas.

Mas, como esses princípios podem ser aplicados em uma memória? Este é o objetivo do artigo presente, medir o desempenho e explorar os princípios de localidade espacial e temporal, avaliando o impacto das mudanças do mapeamento de endereços, da quantidade de palavras por bloco, da política de escrita e da política de substituição na arquitetura de uma memória Cache.

Este artigo foi organizado da seguinte forma: Referencial Teórico, cujo descreve os conceitos técnicos utilizados para a configuração das arquiteturas; Trabalhos Correlatos, o qual aborda os artigos relacionados a este, que foram utilizados como embasamento teórico; Metodologia, cuja apresenta os cenários de testes e como foram feitos; Avaliação dos Resultados, a qual aborda uma análise dos resultados obtidos nos cenários definidos e, por fim, a conclusão que apresenta as considerações finais deste artigo.

II. Referencial Teórico

A. Mapeamento Direto

1) Mapeamento Direto: os blocos de memória são organizados em um conjunto. O índice de um endereço seleciona a linha em que a palavra procurada deve estar. Então, a tag desta linha é comparada com a tag do endereço. Mapeamento ilustrado na figura 2.

2) Mapeamento Associativo por Conjunto: os blocos de memória são estruturados em mais de uma via. O índice de um endereço seleciona a linha em que a palavra procurada

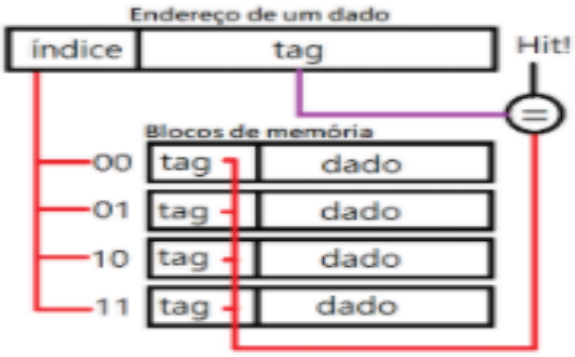

Figura 2. Mapeamento dos Endereços

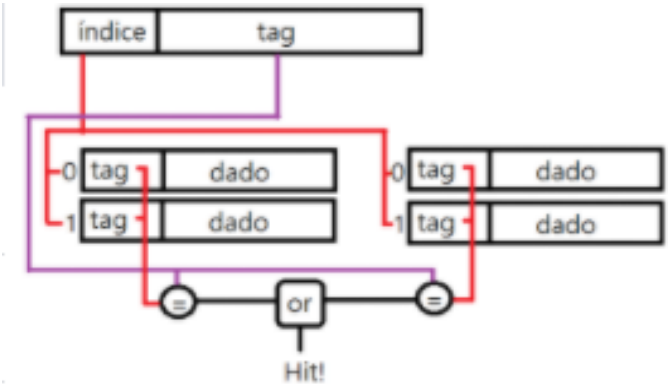

Figura 3. Mapeamento Associativo por Conjunto

deve estar. Então, a tag do endereço é comparada com as tags desta linha em todas as vias. Mapeamento ilustrado na figura 3.

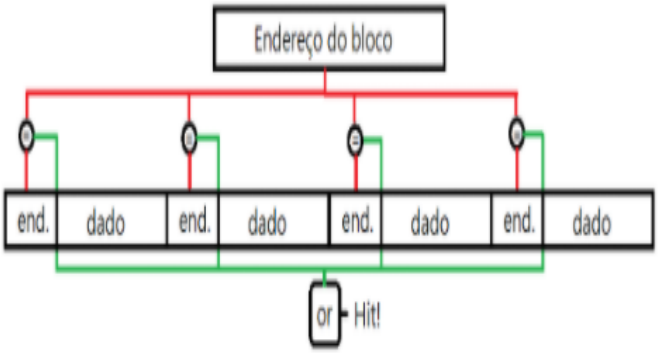

Figura 4. Mapeamento Completamente Associativo

3) Mapeamento Completamente Associativo: neste mapeamento, o número de vias é máximo e os blocos são dispostos na mesma linha. O endereço de uma palavra procurada é comparado com o endereço de todas as vias simultaneamente. É importante observar que o número de comparadores necessários para o mapeamento, é igual ao número de vias da memória, o que torna o mapeamento completamente associativo inviável em muitos casos, já que mais recursos precisam ser utilizados. Mapeamento ilustrado na figura 4.

B. Palavras por bloco

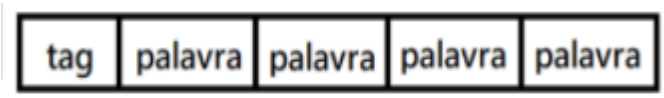

Figura 5. Palavras por bloco

Refere-se à quantidade de palavras que podem ser alocadas em um bloco da memória Cache. As arquiteturas com mais de uma palavra por bloco precisam identificar qual delas está sendo requisitada, por isso, alguns bits do endereço são separados. Na figura 5, quatro palavras podem ser alocadas em um bloco, ou seja, dois bits como offset de bloco são suficientes para identificar a palavra correta.

C. Política de Substituição

Quando uma palavra necessita de um espaço já ocupado na memória, a arquitetura precisa definir uma regra de substituição de dados. As duas políticas de substituição utilizadas neste artigo foram:

- First In First Out: dentre as palavras na memória, a FIFO substitui a que foi alocada primeiro;
- Least Recently Used: dentre as palavras na memória, a LRU substitui a que foi menos utilizada recentemente.

D. Política de Escrita

Quando uma palavra que está na Cache é alterada, esta mesma palavra em níveis hierárquicos inferiores se torna desatualizada e uma regra de escrita precisa definir quando ela precisa ser atualizada. As duas políticas de escritas utilizadas neste artigo foram:

- Write Back: a atualização é replicada aos níveis inferiores somente quando esta palavra é substituída;
- Write Through: a atualização é replicada aos níveis inferiores imediatamente após a alteração da palavra.

E. Outros conceitos

- Hit: quando o dado acessado aparece em algum bloco no nível superior.
- Miss: é se o dado acessado não aparece em algum bloco do nível superior.
- Hit rate: a razão de acessos encontrados pelo número total de acessos ao nível superior.
- Miss rate: é a razão de acessos não encontrados pelo número total de acessos ao nível superior miss ratio = 1 – hit ratio.
- Hit time: é o tempo de acesso ao nível superior da hierarquia de memória, que inclui o tempo necessário para saber se no acesso ocorrerá um hit ou um miss.
- Miss penalty: é o tempo para recolocar um bloco no nível superior e enviá-lo ao processador, quando ocorrer um miss.

F. Simulador de Hierarquia de Memórias Amnesia

Como ferramenta de simulação de memórias, foi utilizado o programa Amnesia, um Software de cunho acadêmico com fins didáticos, que foi desenvolvido por alunos de graduação e da pós-graduação do Instituto de Ciências Matemáticas e de Computação (ICMC) da Universidade de São Paulo (USP). O Amnesia é um simulador de hierarquia de memória de sistemas computacionais. Ele permite simular o comportamento de registradores em um processador, memórias cache, memória principal e memória virtual paginada. Além disso, ele representa as estruturas de hardware e software usadas pela hierarquia de memória, a funcionalidade das mesmas e o impacto no desempenho quando esta hierarquia é utilizada. Diferentes configurações da hierarquia de memória podem ser estabelecidas e comparadas durante as atividades de simulação.

1) Arquiteturas: Para iniciar uma simulação, é necessária a seleção de uma arquitetura contida em um arquivo formatado no padrão xml. Podem ser configurados nesse arquivo campos, como atributos de processador, memória cache, principal e virtual. Para realizar uma simulação com propósito de analisar o funcionamento da memória cache, os atributos do processador, da memória cache e principal, devem ser obrigatoriamente especificados. Desta forma, para o processador deve-se configurar o tamanho da palavra em bytes. Para cada nível da memória cache, deve-se configurar o seu tipo (unificada ou separada), o seu tamanho em palavras, o número de palavras por bloco, o número de blocos por conjunto (i.e. a associatividade), o número de ciclos para cada acesso de leitura, o número de ciclos para cada acesso de escrita, o tempo de cada ciclo, a política de substituição (FIFO ou LRU) e a política de atualização (write-through ou write-back). Para a memória principal, deve-se configurar o seu tamanho em palavras, o número de palavras por bloco, o número de ciclos para cada acesso de leitura, o número de ciclos para cada acesso de escrita e o tempo de cada ciclo.

2) Traces: O arquivo de rastro, que também é chamado de Trace, deve ser carregado. Um arquivo trace é um .txt contendo os acessos de leitura, escrita ou busca realizados na simulação. A estrutura de um arquivo Trace utilizado é composta por uma coluna de comandos, cuja é representada pelo número 1 (escrita) ou 2 (leitura), e uma segunda coluna, que contém os endereços das memórias em Hexadecimal.

III. Trabalhos Correlatos

A. Exploiting Spatial and Temporal Locality of Accesses:A New Hardware-Based Monitoring Approach for DSM Systems [5]

O tema principal abordado neste artigo é a exploração e investigação de localidade espacial e temporal em um Hardware, para o desenvolvimento do artigo o pesquisador inicia introduzindo ao leitor uma das principais aplicações de DSM (Distributed Shared Memory), aplicada em seu projeto The SMiLE PC Hardware. Para experimentos de validação utilizou-se um simulador com monitoramento, coletaram todos os Traces das simulações, executaram programas de Benchmark e assumiram um sistema de DSM com as seguintes propriedades; Latência de acesso de R / W local de 1 ciclo, assumindo que eles podem ser vistos como normalmente armazenados em cache em um sistema normal, Latência de gravação remota de 20 ciclos, correspondendo a aprox. 100 ns em um sistema de 200 MHz, Latência de leitura remota de 1000 ciclos, correspondendo a aprox. 5 us em um sistema de 200 MHz e Latência de bloqueio remoto de 1000 ciclos, pois o processador fica paralisado ao aguardar o resultado da operação remota de leitura-modificação-gravação atômica.

B. O IMPACTO DA HIERARQUIA DE MEMÓRIA SOBRE A ARQUITETURA IPNOSYS,ALEXANDRO LIMA DAMASCENO [6]

O artigo criado pelo Pós graduando em Ciências da computação, Alexandro Lima Damasceno, desempenhou um papel fundamental para a correlação dos temas e estudos propostos para o trabalho, uma vez que para introduzir e ambientar o leitor, o autor precisou utilizar teorias de arquiteturas computacionais como o principal referencial teórico. Logo, conceitos como Hierarquia de Memórias, Classificação das Memórias, Tipos de Memória e comandos de Linguagem de Máquina foram introduzidos ao leitor. O foco de seu artigo foi ”Desenvolver modelos de hierarquia de memória diferentes, destacando suas características e avaliando seus impactos sobre a arquitetura IPNoSys”, tal arquitetura utiliza redes-em-chip para interconectar suas unidades de processamento e roteamento.

C. Amnesia um Objeto de Aprendizagem para o Ensino de [7]

Este artigo evidencia conceitos importantes do simulador de memórias Amnesia, além de mostrar sua importância ao aprendizado sobre memórias. O objeto de aprendizagem Amnesia, caracteriza-se por ser um simulador de hierarquia de memória na arquitetura de Von Neumann. Sua função é demonstrar o funcionamento dos registradores de um processador, memórias caches, principal e virtual de forma didática, com o objetivo de facilitar e melhorar a aprendizagem na disciplina de Organização e Arquitetura de Computadores.

D. The Locality Principle [4]

Como dito por Peter J Denning Pós-graduando da Escola Naval de Monterey e escritor do artigo dito cujo, ”A localidade é um comportamento universal de todos os processos computacionais: eles tendem a se referir repetidamente a subconjuntos de seus recursos em intervalos de tempo estendidos ”, este abstract apresenta uma noção do que o artigo se propõe a trabalhar ao longo da pesquisa tais como aspectos, fundamentos, mecanismos e história dos Princípios da Localidade no mundo da computação. Equações matemáticas são utilizadas para explicar um dos conceitos essenciais para o entendimento de como um computador se comporta sem precisar ligá-lo ou explorar do seu processamento para entendê-lo e tal conceito são os Modelos de Localidade. Para concluir

exemplos abordados pelo autor que foram conhecimentos essenciais para nosso trabalho; ”Existem dois aspectos de localidade: Localidade temporal significa que as referências aos mesmos objetos são agrupadas no tempo e Localidade espacial significa que objetos próximos uns dos outros tendem a ser referenciados juntos”. Além das aplicações técnicas Peter se propõe a contextualizar historicamente como surgir este conceito, ”A localidade está entre os princípios de sistemas mais antigos da ciência da computação. Ele foi descoberto em 1966 durante os esforços para fazer os primeiros sistemas de memória virtual funcionarem bem ”.

IV. Metodologia

Para a realização dos testes, uma arquitetura base de memória foi definida contendo uma RAM com tamanho de 64 palavras e uma Cache de 32 palavras. Também, foram estabelecidos quatro cenários de testes para avaliar o efeito dos princípios de localidade, são eles:

- Mudança no mapeamento de endereços;
- Mudança na quantidades de palavras por bloco;
- Mudança na política de substituição;
- Mudança na política de escrita.

	CACHE	RAM
SIZE	32	64
ASSOCIATIVIVIDADE	1 => 32	-
PALAVRAS / BLOCO	1 => 32	1 => 32
ESCRITA	WB ou WT	-
SUBSTITUIÇÃO	LRU ou FIFO	-

Figura 6. Arquitetura Base

Um dos principais desafios enfrentados foi a construção de modelos de Traces, que atendiam ao objetivo do artigo. Então, para auxiliar na geração dos arquivos, foi implementado em linguagem C, um código que auxilia na geração randômica dos endereços de memórias [2]. Como resultados obtivemos 3 Traces utilizadas nas simulações, são elas:

- Trace Aleatória: contém, inicialmente, escritas sequenciais nos endereços 1 à 64 e, após, instruções de busca em endereços aleatórios;
- Trace Sequencial 8: contém, inicialmente, escritas sequenciais nos endereços 1 à 64 e, após, blocos de 8 instruções sequenciais de busca;
- Trace Sequencial 16: contém, inicialmente, escritas sequenciais nos endereços 1 à 64 e, após, blocos de 16 instruções sequenciais de busca.

Por fim, as métricas coletadas para avaliar o resultado foram o Hit Rate (mede o percentual de acertos na Cache) e o Total Time (métrica que calcula o custo dos acessos de escrita e leitura das memórias).

V. Avaliação dos Resultados

A. Cenário 1: Mudança no mapeamento de endereços

Arquitetura do cenário: Cache com tamanho 32, Write Back como política de escrita e LRU como política de substituição. A quantidade de palavras por bloco e o mapeamento são alterados durante a execução dos testes. O resultado pode ser visto na figura 7.

HIT RATE: Mapeamentos x Palavras/Bloco (Trace Blocos de 16 Aleatórios)						
Palavras/ Bloco	Direto	Conj.Ass. (2 vias)	Conj.Ass. (4 vias)	Conj.Ass. (8 vias)	Conj.Ass. (16 vias)	Completa mente ass.
1	0,4503	0,4556	0,3687	0,3191	0,3049	0,3173
2	0,7039	0,695	0,6578	0,6382	0,6436	-
4	0,8351	0,828	0,8049	0,8031	-	-
8	0,8989	0,8882	0,8865	-	-	-
16	0,9379	0,9255	-	-	-	-
32	0,9379	-	-	-	-	-

Figura 7. HIT RATE: Mapeamentos x Palavras/Bloco (Trace Sequencial 16)

Nos resultados obtidos na tabela acima, foi observado que o aumento da associatividade impacta diretamente no hit rate. Já que, na medida em que a associatividade se aproxima do total, o hit rate registra um ápice e logo depois começa a diminuir. Isto significa que após o ponto crítico, o ganho na localidade espacial não compensa o alto Miss penalty decorrente da falta de linhas, e então, a localidade temporal é comprometida. A figura 8 ilustra em formato de gráfico que o resultado também é obtido nas demais traces e de forma mais acentuada na ‘Trace Sequencial 16’:

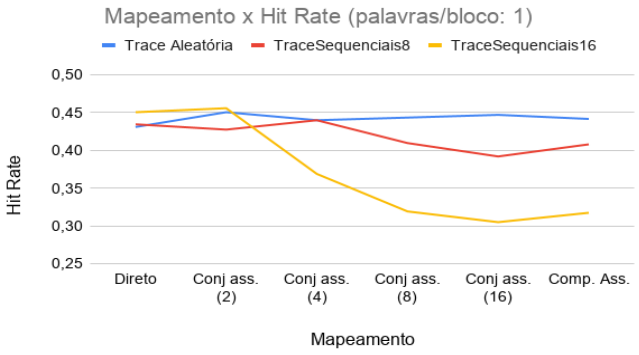

Figura 8. Mapeamento X HIT(palavras por bloco:1)

B. Cenário 2: Mudança na Quantidades de Palavras por Bloco

Arquitetura do cenário: Cache com tamanho 32, Write Back como política de escrita, LRU como política de substituição e com mapeamento direto. A quantidade de palavras por bloco

HIT RATE: Traces x Palavras/Bloco			
Palavras/ Bloco	Trace Aleatória	Trace Sequencial 8	Trace Sequencial 16
1	0,4308	0,4343	0,4503
2	0,4414	0,6826	0,7039
4	0,5177	0,7996	0,8351
8	0,5354	0,8617	0,8989
16	0,5514	0,9131	0,9379
32	0,5478	0,9148	0,9379

Figura 9. HIT RATE: Traces x Palavras/Bloco

é alterada durante a execução do cenário. O resultado pode ser visto na figura 9.

No cenário 2, foi investigada a relação entre a quantidade de palavras por bloco e o Hit Rate. O resultado obtido neste cenário de teste foi satisfatório e mostra que a quantidade de palavras por bloco também impacta diretamente no Hit Rate da memória Cache. O aumento da quantidade de palavras em um bloco aumenta o Hit Rate, porque também favorece a exploração da localidade espacial. Mas como mostrado anteriormente no cenário 1, o miss penalty também aumenta e em certo ponto a localidade temporal é comprometida. Neste teste, foi possível constatar o efeito negativo do aumento de palavras por bloco somente na Trace Aleatória, isso possivelmente se deve ao tamanho curto da memória Cache ou um viés das traces sequenciais. A figura 10 ilustra o resultado obtido.

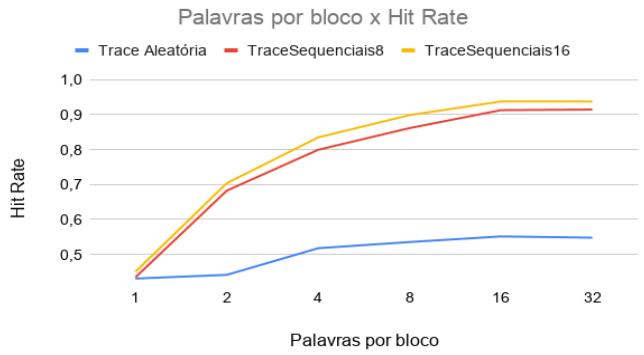

Figura 10. Palavras por bloco X HIT

C. Cenário 3: Mudança na Política de Substituição

Arquitetura do: Cache com tamanho 32, Write Back como política de escrita e uma palavra por bloco. A política de substituição e o mapeamento foram alterados durante a execução do cenário. O resultado pode ser visto na figura 11.

Os valores obtidos neste cenário evidenciam que a escolha da política de substituição também impacta diretamente no Hit Rate da memória Cache e que a LRU teve um desempenho melhor comparada à FIFO. A explicação devida a este

HIT RATE: Política de Substituição x Mapeamento						
	Trace Aleatória		Trace Sequencial 8		Trace Sequencial 16	
Associativ idade	FIFO	LRU	FIFO	LRU	FIFO	LRU
Direto	0,4308	0,4308	0,4343	0,4343	0,4503	0,4503
Conj ass.(2)	0,4255	0,4503	0,4255	0,4273	0,4609	0,4556
Conj ass.(4)	0,4361	0,4397	0,4503	0,4397	0,3794	0,3687
Conj ass.(8)	0,4273	0,4432	0,3918	0,4095	0,3404	0,3191
Conj ass.(16)	0,4326	0,4468	0,3989	0,3918	0,3244	0,3049
Comp. Ass.	0,4361	0,4414	0,39	0,4078	0,3226	0,3173

Figura 11. HIT RATE: Política de Substituição x Mapeamento

resultado é que a política Least Recently Used preserva na memória as palavras mais recentemente utilizadas, ou seja, explora desta forma o princípio da localidade temporal. Já a First In First Out, apenas substitui a primeira palavra alocada na memória, sem considerar se esta palavra foi utilizada recentemente. Também, é válido ressaltar que no mapeamento direto não há política de substituição, por isso o Total Time deste mapeamento consta como igual na FIFO e na LRU. A figura 12 ilustra o resultado encontrado com a Trace Aleatória:

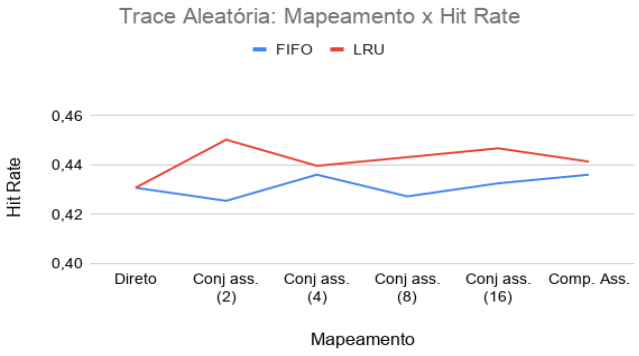

Figura 12. Trace aleatória: mapeamento X HIT

D. Cenário 4: Mudança na política de Substituição

Arquitetura do cenário: Cache com tamanho 32, mapeamento por conjunto associativo de 2 vias e LRU como política de substituição. A política de escrita e a quantidade de palavras por bloco foram alteradas durante a execução do cenário. O resultado pode ser visto na figura 13.

No cenário 4, foi investigado como a política de escrita interfere no desempenho da memória. Os resultados gerados mostram que essa configuração não tem impacto no Hit Rate da memória Cache. Entretanto, houve impacto no Total Time. Portanto, o uso da regra Write Back foi mais eficiente comparado ao Write Through, já que apresentou um Total Time inferior. Este resultado não evidencia uma exploração dos princípios de localidade, mas mostra uma otimização significativa dos acessos à memória. Além disso, na arquitetura com uma palavra por bloco, o Total Time do Write Back e Write Through, curiosamente foram iguais em todas as Traces,

TOTAL TIME: Política de Escrita x Palavras/Bloco						
	Trace Aleatória		Trace Sequencial 8		Trace Sequencial 16	
Palavras/ Bloco	WT	WB	WT	WB	WT	WB
1	5008	5008	5138	5138	4978	4978
2	4858	4218	3678	3038	3628	2988
4	4618	3658	3048	2088	2878	1918
8	4398	3278	2688	1568	2538	1418
16	4428	3228	2518	1318	2328	1128

Figura 13. TOTAL TIME: Política de Escrita x Palavras/Bloco

este fato ainda não foi explicado. A figura 14 ilustra o resultado do cenário utilizando a "Trace Sequencial 8":

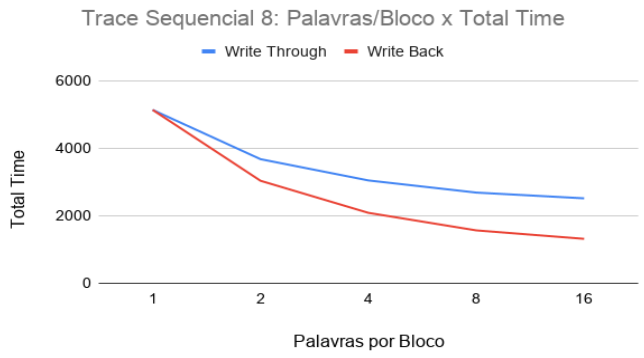

Figura 14. Trace Sequencial 8: Palavras/Bloco x Total Time

VI. Conclusões

Este artigo teve como objetivo demonstrar os impactos da alteração do mapeamento de endereços, da quantidade de palavras por bloco, da política de escrita e da política de substituição no desempenho da memória Cache e nos princípios de localidade.

De modo geral, os resultados obtidos foram satisfatórios e esperados. Foi possível observar que a alteração da associatividade e quantidade de palavras por bloco possui relação direta com o princípio de localidade espacial e temporal.

Já a política de substituição, influencia somente no princípio de localidade temporal. E a política de escrita não possui relação com os princípios, mas tem efeito no desempenho e eficiência da memória.

Além disso, foi possível constatar que as Traces de teste podem conter vieses e influenciar no resultado das simulações. Inicialmente, o intuito era proporcionar traces de melhor caso, caso médio e pior caso. Entretanto, o que foi demonstrado é que a "Trace Aleatória", em todos os cenários, teve variação menor no Hit Rate do que a "Trace Sequencial 8" e menor ainda na "Trace Sequencial 16". Uma possível explicação deste fato é que instruções de busca sequenciais são mais impactadas pelas mudanças na exploração da localidade temporal e espacial.

Em trabalhos futuros, novos experimentos serão planejados com o intuito de atenuar vieses nas traces e investigar melhor o impacto negativo do excesso de palavras por bloco (simulado no cenário 2 da seção Avaliação dos Resultados). Além de examinar o motivo do Total Time ser igual, utilizando Write Through ou Write Back em uma arquitetura com uma palavra por bloco (simulado no cenário 4 da seção Avaliação dos Resultados).

Por fim, as tabelas que registram todos os resultados obtidos, a arquitetura base, as traces utilizadas e o código gerador de traces serão disponibilizados em um repositório do GitHub. [8]

Referências

[1] TIOSSO, F. ; Bruschi, S.M. ; SOUZA, P. S. L. ; BARBOSA, E. F. . Amnesia: um Objeto de Aprendizagem para o Ensino de Hierarquia de Memória, Proceedings of the 25o. Simpósio Brasileiro de Informática na Educação (SBIE 2014), Dourados, Sociedade Brasileira de Computação, 2014. v. 1. p. 1-10.

[2] https://replit.com/@ArthurB70/GeraTrace#main.c

[3] Carlos Emílio de Andrade Cacho, "Desenvolvimento e utilização de recursos educacionais abertos para colaborar com o ensino de memória virtual".

[4] Denning, Peter J., The Locality Principle, Monterey, Faculty and Researcher Publications.

[5] HOCKAUF, Robert; KARL, Wolfgang; LEBERECHT, Markus; Exploiting Spatial and Temporal Locality of Accesses: A New Hardware-Based Monitoring Approach for DSM Systems, Munchen: Universitt Mnchen.

[6] LIMA, Alexandre; Impacto da Hierarquia de Memória sobre a arquitetura IPNOSYS; Mossoró; Universidade do Estado do Rio Grande do Norte.

[7] TIOSSO, Fernando; MAZZINI, Sarita; BARBOSA, Ellen; Amnesia: a Learning Object for Memory Hierarchy Teaching; Pesquisa Acadêmica; São Carlos, Brazil.

[8] https://github.com/ArthurB70/ArtigoHierarquiadeMemorias