Simulador Amnesia: Um estudo sobre o impacto de diferentes técnicas e políticas em um computador

Filipe Arthur Ferreira Silva [1] **, Henrique Augusto** [1] **,**
Letícia Americano Lucas [1] **, Rafael Amauri Diniz Augusto** [1]

[1]Instituto de Ciências Exatas e Informática – Pontifícia Universidade Católica de Minas gerais (PUG-MG)

{fafsilva,henrique.rodrigues,lalucas,radaugusto}@sga.pucminas.br

Resumo. *O presente artigo tem como objetivo apresentar o efeito prático de políticas de substituição e mudanças arquiteturais como técnicas para melhor exploração das localidades temporal e espacial em uma dada arquitetura de computador. Para este fim foram designados diferentes testes que avaliam diferentes partes da arquitetura e quantificam seus impactos para mensurar o impacto em performance de uma melhor exploração da localidade temporal e da localidade espacial.*

1. Introdução

Trabalhando com hierarquia de memória, descobre-se a necessidade da adaptação pelas mudanças da tecnologia na contemporaneidade. Dessa forma, constitui-se essencial a realização de testes que comprovem a eficácia de determinadas técnicas e práticas a fim de concretizar-se um ganho de performance nos computadores. Sabendo do impacto que a memória possui no que diz respeito ao desempenho e processamento de dados, quais são as melhores técnicas até que ponto é possível quantificar essa melhoria?

Neste artigo buscamos, através de diferentes testes, trazer comparações entre diferentes técnicas e métodos de gerenciamento e distribuição da memória a fim de quantificar a melhora trazida por cada um, e após isso selecionar a configuração que mais traz melhorias de desempenho. Este artigo foi organizado com estes testes sistemáticos em mente, a fim de aproveitar ao máximo as funcionalidades do simulador Amnesia que comprovem o impacto das localidades temporal e espacial.

2. Trabalhos correlatos

2.1. "Amnesia: um Recurso Educacional Aberto para o Ensino de Memória Virtual", Instituto de Ciências Matemáticas e de Computação – ICMC/USP, 2016.

Este artigo entra em detalhes a fundo sobre o funcionamento do simulador Amnesia, com um foco especial em sua utilidade como ferramenta de aprendizado sobre hierarquia e gerenciamento de memória. O artigo apresenta também testes que comprovam que alunos que utilizam o Amnesia acertam mais questões sobre arquitetura de computadores em provas, configurando o Amnesia como uma ferramenta valiosa. Enquanto o foco do artigo é apresentar o Amnesia como uma ferramenta educacional, o foco do nosso artigo é apresentar o Amnesia como uma plataforma de testes para comprovar os benefícios das localidades temporal e espacial (CACHO et al, 2016).

2.2. Hierarchical Memory System Environment (HSME)

O HSME é um simulador de gerenciamento de memória e hierarquia de memória com foco em TLB, memória principal e memória cache, com os três módulos funcionando de maneira isolada. Esse funcionamento de maneira isolada foi identificado pelo grupo como sendo uma limitação do simulador, visto que isso impede a verificação do impacto de mudanças em um componente ser traduzida em um impacto em outro componente diferente. Naturalmente, por conta dessa impossibilidade de realizar os mesmos testes com o mesmo nível de profundidade, o HSME foi identificado pelo grupo como uma versão mais simplificada do Amnesia (DJORDJEVIC et al, 1998).

2.3. ParaCache

ParaCache é um simulador desenvolvido pela Nanyang Technological University que possui a capacidade de simular interações com cache e memória virtual. As interações oferecidas são: Direct Mapped Cache, Fully Associative Cache, 2-Way SA, 4-Way e Virtual Memory. O simulador mostra o funcionamento das operações com animações e também possui uma seção de Knowledge Base, que serve como documentação do projeto (NTU, 2021). Embora próximo, o ParaCache ainda não possui tantas funcionalidades quanto o Amnesia, e isso faz alguns dos testes apresentados no presente trabalho serem impossíveis de fazer, como os testes envolvendo a alteração de associatividade da cache.

3. Metodologias

A arquitetura base utilizada consiste nas seguintes especificações utilizando o simulador Amnesia (USP, 2021):

- Trace: Tamanho de palavras de 4 bytes.
- CPU: Tamanho de palavras de 4 bytes.
- Memória Principal: Tamanho de bloco de 2 bytes, tamanho da memória de 16 bytes, 1 ciclo por leitura, 2 ciclos por escrita, tempo de ciclo de 10ns.
- Cache: Unificada, 1 ciclo por leitura, 1 ciclo por escrita, tempo de ciclo de 1ns, tamanho de memória de 8 bytes, associatividade 2, política de escrita WriteThrough e política de substituição FIFO.
- Memória Virtual: Tamanho de página 4, tamanho de disco de 16 bytes, 1 ciclo por leitura no disco, 2 ciclos por escrita no disco, tempo por ciclo de 100ns, política de substituição para tabela de páginas FIFO.
- TLB: Unificada, tamanho de memória 2, 1 ciclo por leitura, 2 ciclos por escrita, tempo por ciclo de 1ns, política de substituição FIFO.

O intuito por trás de escolher estes componentes é formar uma base forte que permita monitorar completamente o efeito de mudanças arquiteturais e de tomada de decisão no projeto.

Tendo isto em vista, foram feitos testes avaliando mudanças nas políticas de substituição, alteração da TLB, níveis de associatividade e separação da memória cache.

4. Avaliação dos resultados

A fim de verificar os benefícios de localidade temporal e espacial, foram feitos testes avaliando a correlação entre alterações em diferentes componentes e as melhorias em desempenho associadas a elas, levando em conta os conceitos de localidade temporal e espacial.

4.1. Políticas de Substituição

A memória de um computador possui diferentes níveis, seguindo uma hierarquia (TOY & ZEE, 1986). A fim de melhor utilizar cada um dos níveis e maximizar a reutilização de dados, diferentes políticas de substituição em cada um dos níveis da memória podem ser utilizadas. Para tal discussão, é relevante conceituar as duas políticas que foram utilizadas no presente trabalho: FIFO, que consiste em substituir dados em uma espécie de fila - sempre removendo o primeiro a ser inserido -, e LRU, que consiste em remover sempre o dado menos recentemente utilizado.

Também é importante explicitar que estas a seguir são técnicas de gerenciamento de memória, e não afetam o tamanho ou distribuição física e arquitetural de qualquer forma. Para verificar os efeitos da FIFO e da LRU, a arquitetura base (que utiliza o algoritmo FIFO) foi alterada para utilizar a política LRU em todos seus componentes.

Na figura 1 podemos ver o impacto da implementação das duas políticas na memória principal.

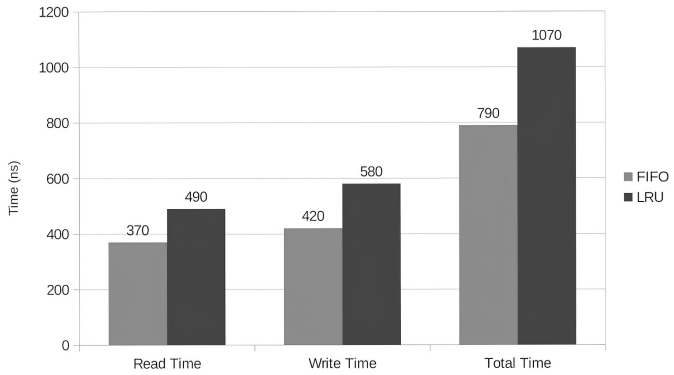

Figura 1. Impacto no tempo de leitura, escrita e tempo total da implementação das políticas FIFO e LRU na memória principal.

Na figura 1 podemos ver que a política FIFO apresenta uma melhora de aproximadamente 35% no tempo total em relação à política LRU. Isso se deve ao algoritmo LRU favorecer muito a localidade temporal dos dados, e o conjunto de dados de entrada não reflete essa necessidade por localidade temporal. Ainda, à medida que o volume de dados aumenta, a LRU desfavorece os dados mais antigos e isso faz com que seja necessário buscá-los na memória principal. Isso explica o tempo gasto na memória principal ser tão maior para o algoritmo LRU em comparação ao FIFO.

Na figura 2 pode ser vista uma situação parecida: o impacto da política LRU na cache também é detrimental, confirmando que a localidade temporal não traz melhora ao conjunto de dados de entrada utilizados.

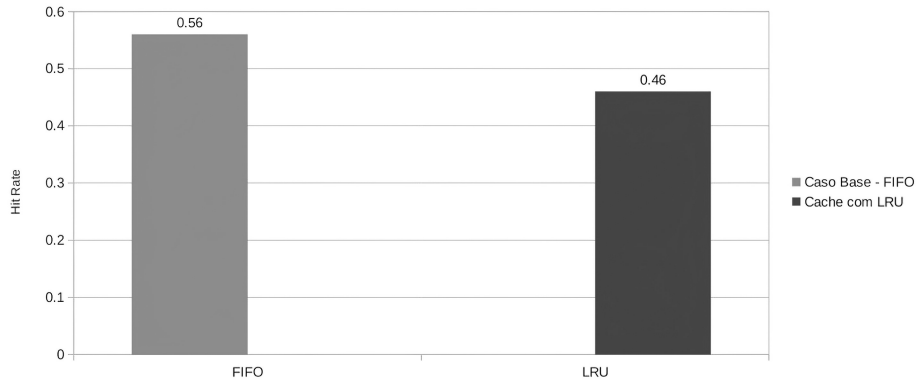

Figura 2. Impacto no hit rate após a implementação das políticas FIFO e LRU na memória cache.

Analisando a figura 2, é mostrado que o hit rate diminui em 10% quando é feita uma mudança da política de substituição de FIFO para LRU. Isso expande o que foi explicado na figura 1, e mostra como explorar a localidade temporal não trouxe melhoras nos resultados do caso de testes.

Analisando, agora, o impacto da LRU no disco:

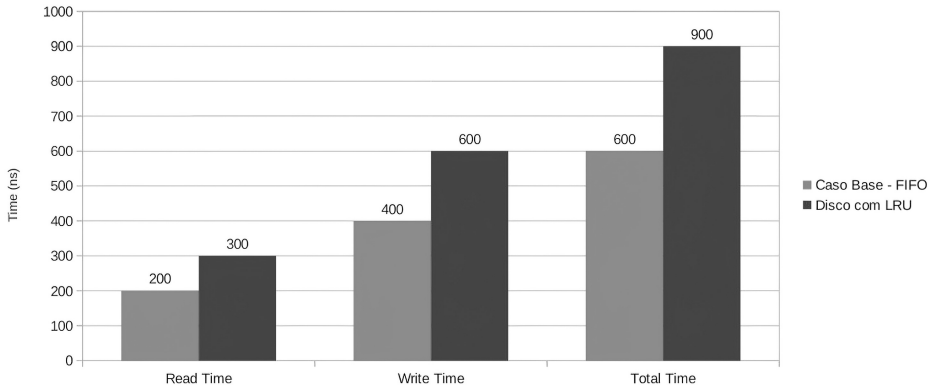

Figura 3. Impacto no tempo de leitura, escrita e tempo total da implementação das políticas FIFO e LRU no disco.

Na figura 3 pode-se ver os efeitos da LRU no disco, embora os números aqui mostram uma piora maior em relação à FIFO do que antes. Pode-se notar uma piora de 50% em todas as métricas após a mudança para a política LRU. Vale a pena lembrar que cada acesso ao disco demora 100ns na arquitetura do presente trabalho, logo, na realidade, foram feitos um acesso de leitura e dois de escrita a mais após a implementação da política LRU.

Por mais que existam diferentes técnicas para tirar proveito da localidade temporal, a política FIFO apresentou resultados melhores que a alternativa LRU em todos os diferentes componentes da arquitetura. Isso também é corroborado por estudos recentes comparando ambas técnicas (EYTAN et al, 2020).

4.2. TLB

A TLB pode ser visualizada como uma memória cache da tabela de páginas. O seu principal objetivo é evitar acessos à tabela de páginas, e por isso ela também armazena dados. Naturalmente, ela também está sujeita a políticas de substituição para um gerenciamento de memória mais eficiente (TANENBAUM & BOS, 2014). No presente conjunto de testes, iremos explorar o conceito de localidade espacial ao aumentar o tamanho da TLB.

Para verificar estes resultados, foi utilizada a mesma arquitetura, mas alterando o tamanho da TLB.

A figura 4 apresenta o primeiro teste:

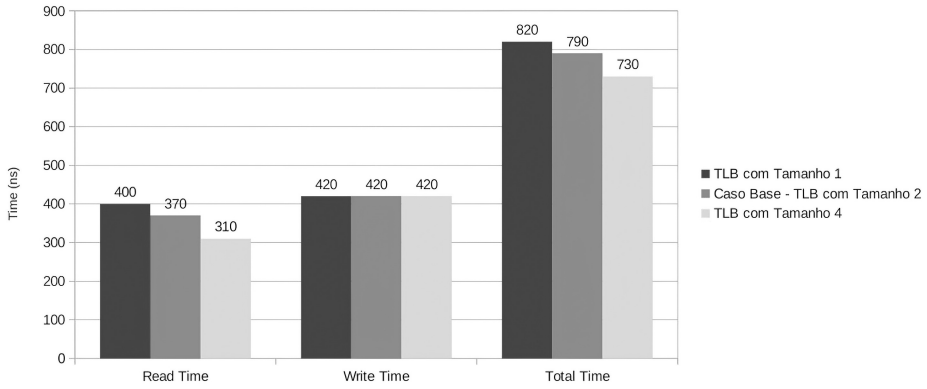

Figura 4. Impacto no tempo de leitura, escrita e tempo total da implementação da alteração do tamanho da TLB na memória principal.

Na figura 4 são apresentados 3 casos: A arquitetura base, que apresenta tamanho de 2 bytes, a arquitetura base modificada para ter uma TLB com o dobro do tamanho (4 bytes) e a arquitetura base modificada novamente para ter uma TLB com metade do tamanho. Pode-se notar que o aumento da TLB causa uma diminuição de aproximadamente 8% no tempo total gasto em operações da RAM, e também uma diminuição em 19% no tempo gasto com operações de leitura. Isso se deve por uma TLB maior fazer menos acessos à tabela de páginas, o que por extensão garante menos acessos à memória principal e ao disco. Da mesma forma, uma TLB com metade do tamanho apresenta uma piora de 8% no tempo gasto com leituras na RAM e uma piora de 4% no tempo total gasto na memória principal. Isso mostra os benefícios de uma TLB maior para tirar um proveito maior da localidade espacial, bem como da possibilidade de se poder armazenar mais dados.

Outra forma de ilustrar esses resultados é com uma comparação de acessos à tabela de páginas.

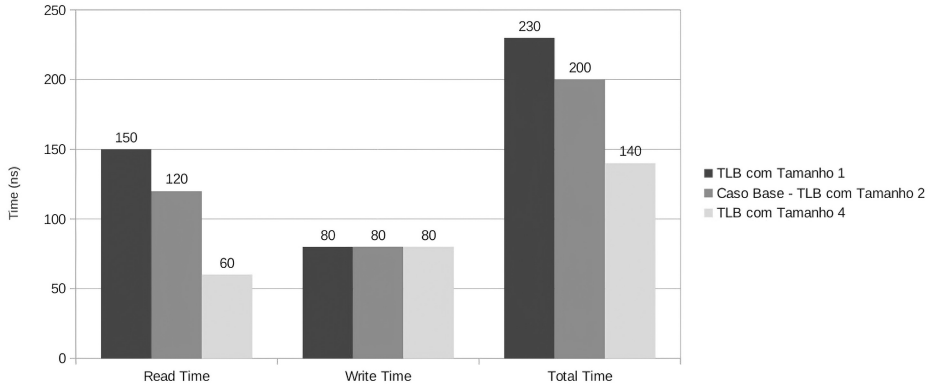

Figura 5. Impacto no tempo de leitura, escrita e tempo total da implementação da alteração do tamanho da TLB na tabela de páginas.

Na figura 5 é possível ver uma confirmação de o que foi exposto antes, com o resultado de outro teste. Na diminuição do tamanho da TLB, houve uma piora de 25% no tempo gasto com operações de leitura e também uma piora de 15% no tempo total gasto. Agora, com o aumento do tamanho da TLB, houve uma diminuição de 50% no tempo de operações de leitura e uma diminuição de 30% no tempo total, o que comprova o efeito da localidade espacial em trazer uma melhora. Com o aumento da TLB, são feitos menos acessos de leitura à tabela de páginas, o que significa menos acessos à memória principal e ao disco.

4.3. Associatividade

A associatividade em um projeto de arquitetura tem grande impacto no desempenho, visto que nela é feita a distribuição do espaço de como os dados são armazenados, o que faz com que hits e misses ocorram em locais diferentes, dependendo de como a arquitetura é estruturada.

No teste apresentado para esta sessão, serão comparados o mapeamento direto (uma via), a associatividade por conjunto (duas vias) e a associatividade totalmente associativa (quatro vias), a fim de avaliar com outra métrica o impacto da localidade espacial no desempenho de um projeto de arquitetura de computador.

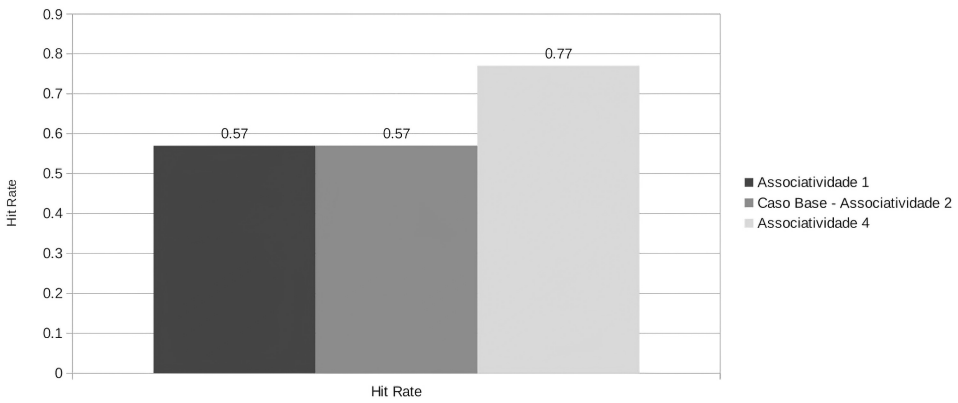

Figura 6. Impacto no hit rate após a alteração da associatividade na cache.

Na figura 6 é possível perceber que houve uma mudança significativa no hit rate da arquitetura após a mudança da associatividade para ser totalmente associativa. Isso se deve ao fato de a cache ser totalmente associativa tirar proveito máximo das quatro vias e maximizar o efeito da localidade espacial, trazendo assim uma melhora significativa para a arquitetura. Também pode ser visto que o mapeamento direto não trouxe piora para o desempenho, como seria esperado. A hipótese que o grupo traz para esse fenômeno é uma dependência séria por parte da arquitetura na localidade temporal ao invés da espacial, ou seja, o desempenho da cache nessa arquitetura está pouco ligado à localidade espacial, o que faz com que seja preciso uma arquitetura com alta associatividade para que seja refletida uma diferença.

A mesma situação pode ser vista na figura 7, quando é alterada a associatividade na memória principal, o que também traz uma melhora significativa.

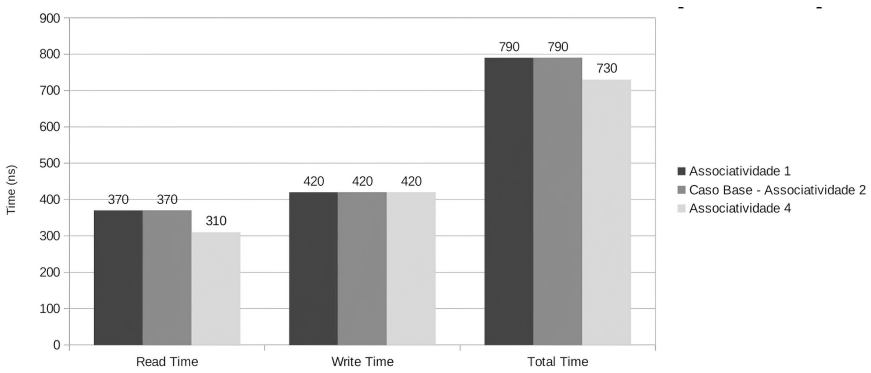

Figura 7. Impacto no tempo de leitura, escrita e tempo total após alteração da associatividade na memória principal.

Na figura 7 é observado um resultado semelhante: Uma melhora significativa no tempo de leitura gasto na memória principal e no tempo total após a alteração para associatividade total. Da mesma forma, não houve piora nem melhora após a mudança para mapeamento direto, o que também reflete uma limitação alta da associatividade no desempenho e na localidade espacial.

4.4. Separação da Cache

A fim de verificar o impacto de uma cache separada na performance e como ela permite uma maior atuação para localidade temporal e espacial (BATISTA et al, 2015), foi feita uma separação da cache entre cache de instruções e cache de dados, ambas com o mesmo tamanho - 4 bytes, que é metade do tamanho da cache unificada presente na arquitetura base.

O teste consistiu em atribuir as mesmas configurações e tempos de acesso para as duas caches, bem como as mesmas políticas de escrita e de substituição a fim de a única variável ser a própria separação da cache.

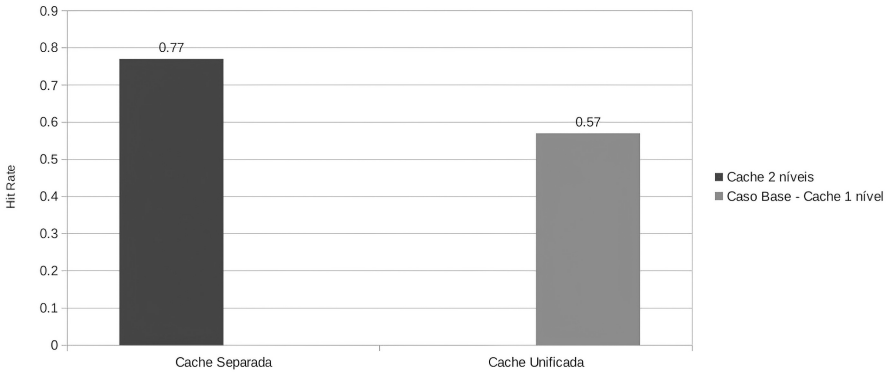

Figura 8. Impacto no hit rate após a separação da cache.

Na figura 8 podemos ver os efeitos de uma cache separada no hit rate da cache. Essa melhora é causada pela separação da cache entre uma cache de instruções e uma cache de dados, o que faz com que substituições ocorram apenas quando necessário. É relevante notar que o tamanho absoluto das duas caches é o mesmo, então essa melhora também pode ser entendida como uma melhor abertura de espaço para a localidade temporal fazer efeito. Por exemplo: em uma cache cheia, se o item removido para uma nova inserção de dado for uma instrução que voltaria logo em seguida, na cache unificada isso resultaria em dois misses, mas na cache separada isso seria apenas um miss, já que o dado substituiria alguém apenas na cache de dados e não na cache de instruções. O reaproveitamento da instrução na cache de instruções é uma manifestação da localidade temporal nessa divisão da cache, o que explica a melhora de aproximadamente 20% no hit rate.

5. Conclusão

No presente trabalho foi utilizada uma arquitetura base que foi modificada de diferentes formas a fim de realizar diferentes testes com o intuito de comprovar os benefícios das localidades temporal e espacial na performance de uma arquitetura, e o grupo conclui que esse objetivo foi atingido. O grupo gostaria de destacar especialmente os resultados obtidos com a separação da memória cache em dois níveis, que foram especialmente influentes na localidade temporal da arquitetura, e uma possível divisão da cache em mais de dois níveis pode trazer inclusive mais melhorias.

A fim de investigar o impacto de outras técnicas e como elas afetam o projeto, o grupo sugere como temas para trabalhos futuros a exploração de políticas de substituição mais complexas, visto que as disponíveis no simulador Amnesia são relativamente simples e podem afetar o desempenho significativamente.

Referências

Batista, A; Farias, L; Segundo, M. A; Barbosa, I. J. M (2015). Caches Multinível. UFRPE.

Cacho, C. E. A; Souza, P. S. L; Bruschi, S. M; Barbosa, E. F. B; Tiosso, F (2016). Amnesia: um Recurso Educacional Aberto para o Ensino de Memoria Virtual. Instituto de Ciências Matemáticas e de Computação - ICMC/USP.

Djordjevic, A; Milenkovic; S. Prodanovic. A hierarchical memory system environment. In Proc. of the 1998 Workshop on Computer Architecture Education, WCAE '98, New York, 1998. ACM.

Eytan, O., Harnik, D., Ofer, E., Friedman, R., & Kat, R.I. (2020). It's Time to Revisit LRU vs. FIFO. IBM Research.

NTU. ParaCache Simulator, 2021. Disponível em: <https://www3.ntu.edu.sg/home/smitha/ParaCache/Paracache/start.html>

Tanenbaum, A. S; H, Bos (2014). Modern Operating Systems. Pearson Education.

Toy, Wing; Zee, Benjamin (1986). Computer Hardware/Software Architecture. Prentice Hall. p. 30. ISBN 0-13-163502-6.

USP. Amnesia Simulator, 2021. Disponívem em: <http://amnesia.lasdpc.icmc.usp.br/amnesia-en/>