Proposta de Ferramenta para o Auxílio no Aprendizado de Memória Cache em Arquitetura de Computadores

Mateus Fogaça[1], André Vargas[1], Cristina Meinhardt[1]

[1]Centro de Ciências Computacionais (C3) - Universidade Federal do Rio Grande (FURG)

Abstract. *Cache memory is a challenging topic upon first introduction. Students must synthesize a significant amount of computer architecture information, mainly reasonably complex mapping and replacement strategies. This work proposes a cache memory simulator environment for teaching the computer cache memory, developed in JAVA with Swing API.*

Resumo. *Memória cache é um tópico desafiador à primeira vista para os estudantes. Eles precisam sintetizar uma quantidade significativa de informação sobre arquitetura de computadores, principalmente sobre estratégias razoavelmente complexas de mapeamento e algoritmos de substituição. Este trabalho propõe um ambiente de simulação de memória cache para a aprendizagem de sistemas de memória cache, desenvolvida em JAVA com a API Swing.*

1. Introdução

É crescente o interesse em melhorar o ensino de conceitos relacionados ao funcionamento interno de processadores em cursos da área de computação. É especialmente interessante permitir aos estudantes construir o seu conhecimento do funcionamento do processador ao mesmo tempo em que internalizam a informação sobre a operação do computador em baixo nível. Estudos demonstram a dificuldade dos estudantes iniciais dos cursos em compreender a organização interna da hierarquia de memória em um sistema computacional, e o papel da memória cache na comunicação entre a Unidade Central de Processamento (CPU) e a memória principal [Grigoriadou et al. 2006].

Para entender estes conceitos, é importante que os estudantes realizem muitos experimentos com diferentes organizações da memória cache adotando sequencias de acessos à memória de execuções de programa reais. Neste sentido, é de grande relevância um ambiente que permita aos estudantes realizar este tipo de experimentos e que seja de fácil utilização e manipulação. Assim, neste trabalho é apresentado um ambiente educacional para o ensino do funcionamento e organização da memória cache em sistemas computacionais visando dar suporte e aperfeiçoar o processo de aprendizado de sistemas computacionais, além de promover o envolvimento ativo e construtivo dos estudantes. A próxima seção apresentará os detalhes de projeto e implementação desta ferramenta.

2. Projeto e implementação da ferramenta

No ambiente educacional proposto é considerado que um sistema de memória cache possui C entradas de memória cache e M entradas de memória principal, onde $C \ll M$. Quando o processador deseja acessar uma posição da memória, ele indica o endereço físico da memória principal. O sistema de memória procura, inicialmente, pelo endereço na memória cache. Se o endereço já se encontra na cache, diz-se que ocorreu um *hit* de

cache, se não, é preciso buscá-lo na memória ocorrendo um *miss* de cache. Além disso, o ambiente auxilia no entendimento da relação entre o endereço físico da memória principal e a linha da cache onde esta posição será alocada, ou seja, o mapeamento realizado. O ambiente permite aos alunos explorar três tipos de mapeamento: Direto, Associativo e Grupo-Associativo[Stallings 2010]. Além disso, neste trabalho os estudantes podem explorar quatro algoritmos de substituição: Aleatório, *First-In First-Out* (FIFO), *Least Frequently Used* (LFU) e *Least Recently Used* (LRU)[Tanembaum 2007].

A principal contribuição deste trabalho é o desenvolvimento de uma ferramenta educacional com ambiente gráfico no qual é possível observar a execução dos algoritmos de substituição e os tipos de mapeamentos, a partir de uma memória e uma lista de acesso informados pelo usuário. O ambiente foi desenvolvido na linguagem Java com a API gráfica *Swing* devido a sua portabilidade. Diferentemente de trabalhos semelhantes[Stefani and Martins 2005] que detalham os dados trabalhados, optou-se por abstrair o conteúdo da memória a simples valores na base hexadecimal com o intuito de não desviar a atenção do usuário das operações da cache.

A memória cache possui operações que independem do algoritmo e mapeamento adotado. Estas operações deram origem a *interface* Cache. A Figura 1 apresenta o diagrama UML deste projeto da ferramenta. A interface Cache possui as operações `size` (retorna o tamanho da cache), `getLine` (retorna uma determinada linha), `findMemoryPos` (procura um endereço de memória na cache), `access` (solicita o conteúdo de memória de um endereço, se este não estiver na cache, é buscado na memória principal) e `run` (recebe e executa uma lista de acesso). Esta interface deu origem a seis classes que implementam as operações com base nos algoritmos de mapeamento e substituição de cache anteriormente apresentados.

Quanto a implementação dos algoritmos de substituição, o algoritmo mais básico é a substituição Aleatória. O FIFO foi implementado acrescentando um ponteiro para a entrada menos recentemente carregada, no LFU cada linha está associada a um contador que indica o número de vezes que a linha foi acessada e no LRU, toda vez que uma linha na cache é acessada, o contador da linha é zerado e todos os contadores das outras linhas são incrementados.

A Figura 2 apresenta a interface amigável modelada para a ferramenta. Esta interface é composta por 3 janelas principais. Na primeira janela (esquerda), o usuário determina as configurações para a simulação, selecionando o algoritmo de substituição de cache e o tipo de mapeamento e setando o tamanho das memórias principal e cache desejados para o experimento. Na janela central são inseridos os dados da memória principal, ou seja, o conteúdo armazenado e a sequência de acesso, determinando a ordem de posições a serem acessadas durante a execução do experimento. Na terceira janela ocorre a simulação, mostrando o mapeamento realizado de acordo com a sequência de acesso passada. A simulação pode ser realizada ainda no modo passo-a-passo, onde é mostrado as alterações realizadas na cache a cada novo acesso a memória principal. Ainda é possível salvar os resultados da simulação, permitindo que estes dados sejam posteriormente analisados e experimentos comparativos entre diferentes arranjos de memória principal e cache possam ser realizados.

3. Trabalhos Futuros

Nas etapas seguintes do trabalho pretende-se desenvolver ferramentas que avaliem e comparem o desempenho entre algoritmos. Também se estudará a possibilidade de versões para dispositivos móveis e testes do ambiente educacional em sala de aula.

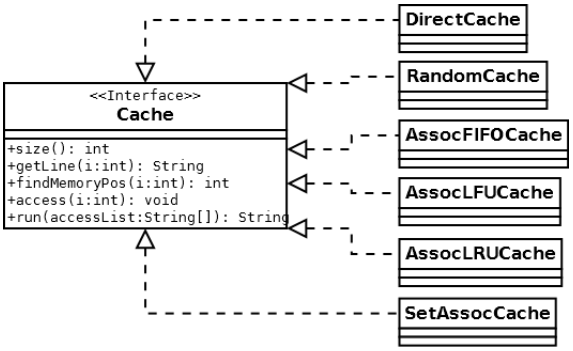

Figura 1. UML do projeto

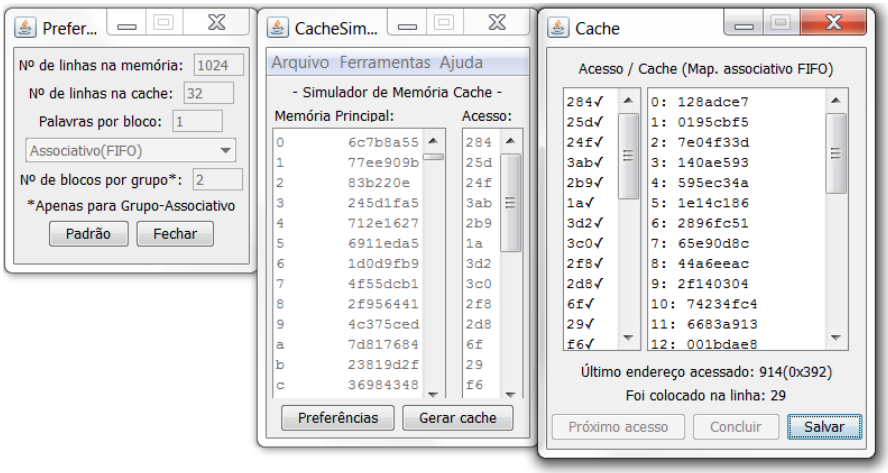

Figura 2. Interface do simulador

Referências

Grigoriadou, M., Kanidis, E., and Gogoulou, A. (2006). A web-based educational environment for teaching the computer cache memory. *IEEE TRANSACTIONS ON EDUCATION*, 49.

Stallings, W. (2010). *Arquitetura e Organização de computadores*. Pearson Prentice Hall, 8a. edition.

Stefani, I. G. A. and Martins, C. A. P. S. (2005). Dcmsim v1.0: Simulador para análise do comportamento e funcionamento estrutural de memórias cache. *VI Workshop em Sistemas Computacionais de Alto Desempenho WSCAD'2005*, pages 230–231. Rio de Janeiro - RJ - Brasil.

Tanembaum, A. S. (2007). *Organização estruturada de computadores*. Pearson Prentice Hall, 5a. edition.