Simuladores de Memória Cache: Um Estudo Comparativo e Sua Aplicabilidade na Educação

Jardel das Chagas Rodrigues[1], Matheus Arleson Sales Xavier[1], Otávio Alcântara de Lima Júnior[1]

[1]*Laboratório de Eletrônica e Sistemas Embarcados (LAESE)*
Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
Av. Contorno Norte, 10 – CEP: 61925-315 – Maracanaú, Ceará
jardel.ifce@gmail.com, matheusarleson@gmail.com, otavio@ifce.edu.br

Resumo

O conceito de memória cache é um tópico importante visto principalmente na disciplina de Arquitetura de Computadores, em cursos da área de computação. Este é um assunto no qual muitos alunos sentem dificuldades de aprendizagem, que podem ser reduzidas com a utilização de um simulador. Este artigo apresenta um estudo comparativo entre alguns dos principais simuladores de memória cache publicados, visando avaliar sua aplicabilidade como ferramenta de apoio ao ensino em cursos introdutórios de Arquitetura de Computadores. Dessa forma, as principais características existentes nos simuladores serão confrontadas, destacando os pontos positivos e negativos de cada um, avaliando a sua empregabilidade para o ensino e fornecendo recursos para que pesquisadores venham a desenvolver novas ferramentas com essas características. O estudo também aponta que um dos principais empecilhos à aplicação destas ferramentas é a dificuldade em localizar estes aplicativos. Finalmente, as características que são desejáveis nos simuladores, bem como a ferramenta que mais se destacou, serão apresentadas, complementando as pesquisas realizadas sobre simuladores de memória cache, concluindo a literatura desse trabalho.

1. Introdução

No decorrer da evolução dos sistemas computacionais, a diferença de velocidade entre a unidade central de processamento e a memória principal se tornou um obstáculo ao aumento do desempenho dos computadores. Uma das formas encontradas para mitigar essa problemática foi o emprego de uma pequena porção de memória veloz e de custo mais elevado, denominada memória cache, localizada entre o processador e a memória principal.

A memória cache armazena temporariamente os dados e instruções que possuem maior probabilidade de serem usados pelo processador, reduzindo o número de acessos à memória principal. A memória cache é largamente empregada nas arquiteturas computacionais modernas. Sua utilização visa à melhora de desempenho do sistema.

O conceito de memória cache está inserido na disciplina de Arquitetura de Computadores. Motivar os alunos a estudar Arquitetura de Computadores pode ser um problema quando o aluno acredita não ser necessário conhecer arquitetura para desenvolver softwares. O desafio em motivar os alunos pode não estar somente na disciplina de Arquitetura de Computadores, mas em todas as disciplinas de um curso, quando estas deixam ausente a correlação de importância com Arquitetura de Computadores [1]. Portanto, devem-se utilizar mecanismos para solucionar essa problemática e tornar a disciplina mais atrativa e menos complexa para os alunos. A utilização de simuladores didáticos contribui para uma melhor compreensão do assunto, uma vez que esses são capazes de traduzir para a realidade os conceitos teóricos através de uma introdução de forma clara e didática dos mesmos [2].

À medida que a complexidade e variedade de sistemas computacionais aumentam, a adequação desses como ferramentas pedagógicas nos cursos de Arquitetura de Computadores diminuem. Em consequência disto, muitos instrutores estão se voltando para simuladores como auxiliares de ensino, muitas vezes empregando o valioso tempo de ensino/pesquisa para construí-los [3]. Um simulador de memória cache pode ser utilizado para facilitar as aulas e auxiliar no aprendizado de sistemas computacionais modernos.

Diversos simuladores já foram propostos e alguns destes serão analisados no decorrer desse trabalho, dentre os quais foram selecionados: KSH [4], Multilevel and Split Cache Simulator (MSCSim) [1], Didactic Cache Memory Simulator (DCMSim) [5], Dinero III [6], Dinero IV [7], CacheSim [8], LBGCache [9] e o SMPCache [10].

O objetivo geral deste artigo é realizar um estudo comparativo mais abrangente, com a inclusão de mais simuladores de memória cache e analisar sua empregabilidade no ensino, com o intuito de facilitar o processo de seleção da ferramenta usada como apoio ao ensino em cursos introdutórios de Arquitetura de Computadores.

O presente artigo é uma versão estendida de um trabalho que promove um estudo comparativo de simuladores de memória cache com um enfoque direcionado ao ensino [11] e diferencia-se do mesmo principalmente pela inclusão de quatro simuladores e pela análise de uma gama maior de características.

O restante do trabalho está organizado em cinco seções. A Seção 2 traz o conceito de memória cache e explica brevemente o seu funcionamento. A Seção 3 do artigo detalha cada simulador. A Seção 4 do artigo exibe os parâmetros de comparação e características desejáveis em um simulador utilizado no ambiente ensino-aprendizagem, bem como um estudo comparativo dos simuladores. A Seção 5 mostra as conclusões deste artigo.

2. Memória cache

No decorrer da evolução computacional, os processadores e outros dispositivos conseguiram um aumento incrível na velocidade de processamento das informações, o que não foi acompanhado pela memória principal, tornando-se lenta e atrasando as demais operações [12]. Na Figura 1, é mostrada claramente essa diferença. Nesse âmbito, precisava-se de uma unidade intermediária que acelerasse as operações que a envolviam. Ao analisar o comportamento dos programas, percebeu-se que o processador tende a referenciar endereços de memória próximos, devido ao grande uso de estruturas de repetições e dados [13].

A memória cache armazena temporariamente os dados e instruções que possuem maior probabilidade de serem usados pelo processador, reduzindo o número de acessos à memória principal.

A Figura 2 mostra o posicionamento da memória cache no sistema, onde a mesma pode ser dividida em vários níveis de acordo com a proximidade da *Central Processing Unit* (CPU), proporcionando uma melhoria no desempenho global.

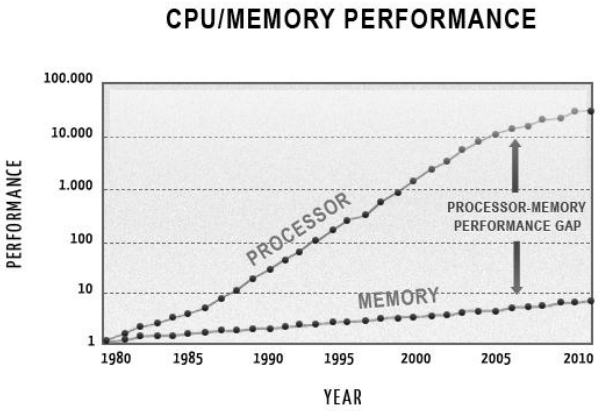

Figura 1. Evolução de desempenho - Memória x CPU [12].

A cache mantém um bloco de informações, composto das posições de memória mais recentemente utilizadas, como também o conteúdo das posições de memória próximas, pois estas possuem alta probabilidade de serem acessadas nas próximas referências de memória.

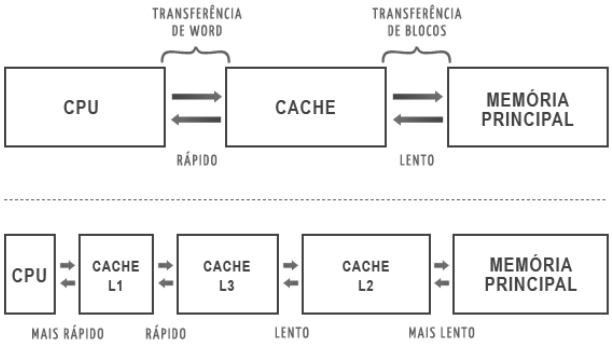

Figura 2. Esquema de uma memória cache e memória principal[13].

Para que esta operação ganhe velocidade, a cache fica próxima ao processador. Apenas quando um dado referenciado não está na cache, um acesso à memória principal é feito.

A Figura 3 apresenta um resumo das operações envolvidas na busca de uma posição de memória em um sistema empregando memória cache.

Embora existam diversas implementações, poucos elementos básicos de projeto servem para classificar e diferenciar as diversas arquiteturas. O primeiro deles: o tamanho da memória cache. Quanto maior, mais caro e rápido, até certo ponto, fica o sistema. Como geralmente a cache é feita do mesmo material do processador, o custo por bit é muito elevado.

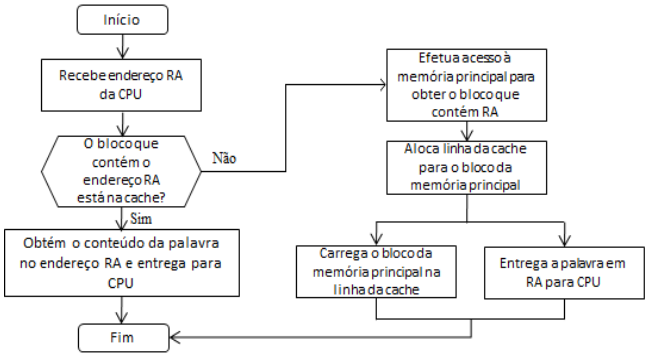

Figura 3. Operação de leitura em memória cache [12].

A função de mapeamento determina como a cache é preenchida, podendo ser de forma direta: cada bloco da memória principal é mapeado para apenas uma linha de cache, associativa: um bloco pode ser alocado em qualquer linha da cache, associativa por conjunto: a cache é dividida em conjuntos, com certo número de linhas cada e um bloco pode estar associado a qualquer linha de um conjunto já pré-determinado.

Os algoritmos de substituição determinam como os dados serão substituídos, podendo ser por diversas técnicas, inclusive: *Least Frequently Used* (LFU), os dados menos frequentemente utilizados serão substituídos; *First In First Out* (FIFO), o primeiro dado a entrar é o primeiro a sair; *Least Recently Used* (LRU), os dados menos recentemente usados são descartados primeiro; ou de forma aleatória.

A política de escrita determina como os dados serão atualizados na memória principal, podendo ser de forma simultânea: *write through* ou os dados são escritos temporariamente na cache e depois na memória, o que denominamos *write back*. Por fim, o tamanho do bloco, que determina a quantidade de dados de um bloco alocado e a quantidade de níveis, para um sistema com vários níveis de cache.

Atualmente a memória cache vem evoluindo e já não está apenas presente nos processadores, mas em quase todos os dispositivos que necessitam dessa aceleração, como os discos-rígidos, por exemplo. Podemos encontrar também sistemas com vários níveis de cache ou caches separadas para dados e instruções.

3. Simuladores

No presente artigo foram escolhidos oito simuladores para realização do estudo, entretanto outras ferramentas podem ser encontradas, como o Prima Cache Simulador [14] e o Loop-way Cache [15], por exemplo. Alguns critérios para a escolha dos simuladores foram adotados, a saber: ou grau de exibição dos simuladores em relação às pesquisas realizadas na *web* sobre o tópico de memória cache e as documentações encontradas referentes aos mesmos, sejam estas artigos, páginas da internet ou manuais.

3.1 KSH

O KSH foi desenvolvido na Pontifícia Universidade Católica de Minas Gerais e desenvolvido na linguagem de programação Java. Esta ferramenta pode ser executada em modo gráfico ou via console, ambos possuindo as mesmas funcionalidades. O KSH implementa as três arquiteturas de memória cache: mapeamento direto; mapeamento associativo e mapeamento associativo por conjunto. Suas políticas de substituição são FIFO e LRU. Possui técnicas de escrita: *write back* e *write through* e o acesso à memória pode ser de modo sequencial ou paralelo.

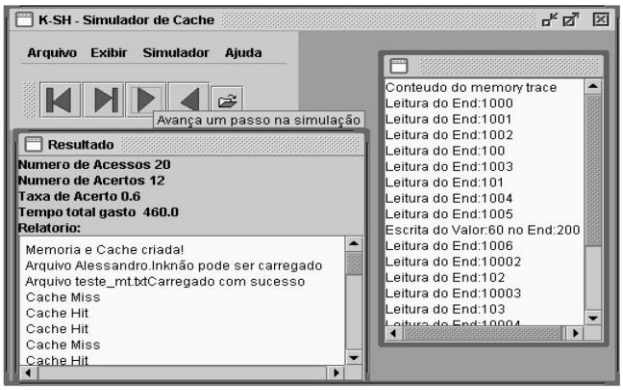

Figura 4. Visão geral do simulador KSH.

O *memory trace* pode ser um arquivo de texto ou um comando via teclado. O programa possui basicamente três ambientes de execução: o primeiro, logo após o *menu* do programa, é o controle da simulação, que permite tanto a observação passo a passo, quanto visualizar diretamente o final da simulação; o segundo, abaixo do controle da simulação, é onde é possível observar as informações; o terceiro, à direita, informa o conteúdo do *memory trace*.

3.2 DCMSim

DCMSim (*Didactic Cache Memory Simulator*) é um simulador que foi projetado na Grupo de Sistemas Digitais e Computacionais (GSDC) na Pontifícia Universidade Católica. Atualmente, a ferramenta possui duas versões. A última versão, selecionada para o estudo, implementa as seguintes arquiteturas de cache: mapeamento direto; mapeamento associativo e

mapeamento associativo por conjunto. Políticas de substituição: FIFO e LRU. Políticas de escrita: *write back* e *write through*. Os dados de entrada são representados pelo conjunto de acessos à memória.

O *memory trace* apresenta-se na forma de um arquivo de texto onde cada linha representa um acesso à memória e tem suas linhas neste formato: <endereço> <indicação de (d)ado ou (i)nstrução> <indicação de (l)eitura ou (e)scrita> <dado/instrução>. Possui um recurso denominado *MissAnalyst* que analisa os acertos e erros dos acessos realizados na memória cache. Na seção denominada *LogBox*, as estatísticas do tempo de acesso são exibidas, tanto da iteração corrente, como de toda a simulação. Isso possibilita gerar um arquivo de relatório contendo a sequência das ocorrências, configurações e os resultados da simulação.

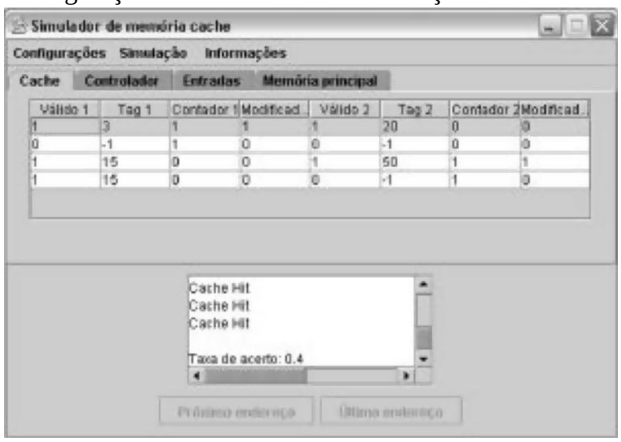

Figura 5. Visão geral do DCMSim.

3.3 MSCSim

MSCSim (*Multilevel and Split Cache Simulator*), foi desenvolvido na Pontifícia Universidade Católica de Minas Gerais. Foi desenvolvido também em Java e implementa as arquiteturas de cache: mapeamento direto; mapeamento associativo e associativo por conjunto. Políticas de substituição: FIFO e LRU. Políticas de escrita: *write back* e *write through*. Acesso à memória: paralelo e sequencial. O programa possui interface gráfica composta de vários ambientes, sendo a janela principal composta de um *menu* principal e um espaço onde é mostrado os arquivos usados recentemente (configurações e *traces*). Seu *memory trace* pode ser feito através de um *trace generator* no programa, o qual permite a criação manual ou randômica seguindo o formato: <endereço> <(L)eitura/(E)scrita> <(I)nstrução/(D)ado>.

Quanto à configuração da arquitetura, pode ser elaborada ou ser importada de um arquivo pré-definido. É possível dividir a cache em vários níveis e diferenciar as características de cada nível. Depois de configurada a arquitetura e informado o *trace*, surge o ambiente de simulação – uma janela com abas: simulação, memória cache, memória principal e estatísticas – a simulação pode ocorrer passo a passo ou diretamente.

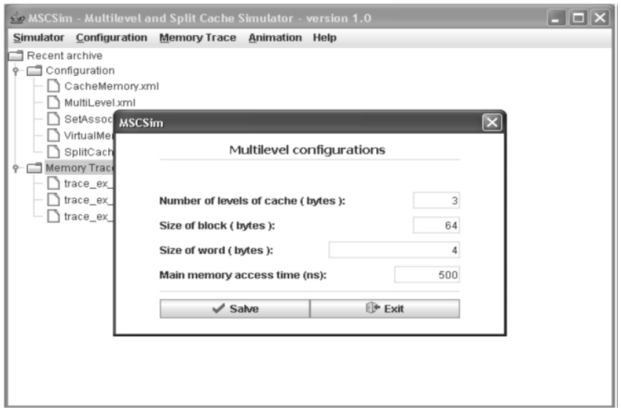

Figura 6. Visão geral do MSCSim.

3.5 Dinero III

O Dinero III foi desenvolvido por Mark D. Hill do Departamento de Ciência da Computação, da Universidade de Wisconsin. Tal simulador é baseado em *trace* com suporte à sub-blocos, também conhecido como localização de setores. A entrada do *trace* é lida pelo simulador no formato *.din.* A simulação orientada a *trace* é frequentemente usada para avaliar a performance de hierarquia de memórias.

Esta versão do simulador não permite realizar simulações simultâneas de múltiplas caches alternativas. A localização de sub-blocos é realizada da seguinte maneira: as *tags* de endereço ainda são associadas com a cache de blocos, porém os dados são transferidos da cache e para a mesma em sub-blocos menores.

O Dinero III possui uma cache dividida para instrução e para dados e uma simulação com cache unificada também é possível, porém não oferece opção para um sistema de cache multi-nível. Possui como políticas de substituição: FIFO; LRU e aleatória. Possui as arquiteturas: mapeamento direto; associativo e associativo em conjunto e, como políticas de escrita, *write back* e *write through*.

Também é possível configurar outras opções, tais como: tamanho do bloco, tamanho da cache de instrução e dados, tamanho dos sub-blocos, políticas e distância da busca, largura do barramento, tamanho da palavra e outros recursos. Não possui interface gráfica e os comandos são inteiramente via console.

3.4 Dinero IV

Desenvolvido pelo Dr. Jan Edler da *NEC Researchs Corporation* e pelo prof. Mark D. Hill do Departamento de Ciência da Computação da Universidade de Wisconsin. O Dinero IV é um simulador de memória cache baseado nas referências do *memory trace* que permite a simulação de caches multi-níveis dentre seus mais diversos tipos, como, por exemplo, uma cache separada para instrução e uma para dados. Possui melhor desempenho e portabilidade do que seu antecessor, o Dinero III.

O Dinero IV não apresenta o tempo decorrido da simulação, sendo a sua principal funcionalidade o cálculo das taxas de acerto e de erro dos acessos feitos à memória cache. Suas políticas de substituição são: FIFO; LRU e aleatória.

Os vários parâmetros de cada cache podem ser configurados separadamente em relação aos fatores como: arquitetura; política e estatísticas. Durante a inicialização é realizada a configuração, uma cache por vez, começando por cada nível. Após a inicialização, os níveis superiores são alimentados a cada referência através da chamada simples de uma função, enquanto que os níveis mais baixos da hierarquia são tratados de maneira automática.

O Dinero IV foi testado nos ambientes: x86/Linux, Alpha/Linux, Alpha/OSF, SGI/IRIX-6, RS6000/AIX, x86/Solaris, e Sparc/Solaris.

```
---Simulation begins.
---Simulation complete.
l1-icache
 Metrics                  Total          Instrn
 -----------------        ------         ------
 Demand Fetches           757341         757341
  Fraction of total       1.0000         1.0000

 Demand Misses             13743          13743
  Demand miss rate        0.0181         0.0181

 Multi-block refs             0
 Bytes From Memory       439776
 ( / Demand Fetches)     0.5807
 Bytes To Memory              0
 ( / Demand Writes)      0.0000
 Total Bytes r/w Mem     439776
 ( / Demand Fetches)     0.5807
```

Figura 7. Algumas métricas obtidas por uma simulação realizada pelo Dinero IV.

3.6 Cachesim

Este simulador foi proposto no Departamento de Engenharia da Informação: Eletrônica, Informática, Telecomunicação, da Faculdade de Engenharia da Universidade de Pisa, por Cosimo Antonio Prete, em 1994, e foi desenvolvido na linguagem de programação C.

O simulador apresenta as arquiteturas de cache: mapeamento direto; mapeamento associativo e mapeamento associativo por conjunto. Políticas de substituição: FIFO; LRU e aleatória. Políticas de escrita: *write back* e *write trough*. Outros recursos da cache também podem ser configurados, como: capacidade; tamanho do bloco e grau de associatividade, por exemplo.

O Cachesim possibilita que o usuário visualize a CPU e as várias atividades da cache durante a execução de uma operação de memória de leitura ou escrita, com objetivo de analisar a desempenho do sistema. A simulação proposta é descrita em três fases: configuração, simulação e análise onde, primeiramente, é escrito um programa em linguagem de programação assembly para que a simulação seja realizada.

Um detalhe relevante desse simulador é a existência de um diagrama representando a memória principal e a cache, que podem ser detalhadas. Por exemplo, para exibir o campo de dados, campo de *tag*, bit de "válido" e de "modificado" de qualquer bloco da cache, basta apenas clicar no ícone da cache e então no ícone do bloco em questão.

O Cachesim também possui três métodos de operação: *single*, *trace* e *exe*. No primeiro modo, o usuário pode configurar a operação de memória especificando o endereço da memória e o tipo de operação. No segundo modo, o usuário pode executar o programa passo a passo, uma operação da memória ou uma instrução por vez.

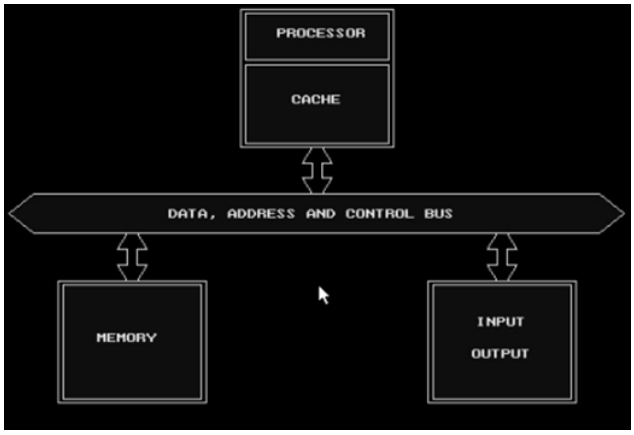

Figura 8. Tela inicial do simulador Cachesim.

Por fim, o último modo possibilita saber pela execução do programa. Após o final da simulação são exibidas três janelas, a primeira mostra as porcentagens das condições de erro e acerto (*hit* e *miss*), entre elas as

condições que ocorrem também durante as operações de busca, operações de armazenamento e operações da pilha, bem como as operações de escrita. Em seguida, os dados são exibidos em forma de gráfico em barras.

A segunda janela exibe sete diagramas: que constituem informações sobre os acessos à memória, erros, acertos, leituras, escritas, entre outros. A última janela exibe um sumário das principais estatísticas, especificando o total de acessos e os acessos de busca, armazenamento e as operações da pilha.

3.7 LBGCache

Simulador desenvolvido utilizando a linguagem Java no CEFET-RN, em 2005. Implementa as arquiteturas de cache: mapeamento direto; associativo e associativo por conjunto. Políticas de substituição: FIFO; LRU; LFU e aleatória. Políticas de escrita: *write back* e *write through*. É possível unificar ou individualizar a cache de dados e de instruções, além de possibilitar a seleção de arquiteturas diferentes para cada uma. Tanto no artigo do simulador quanto no seu manual não foi possível identificar o formato do *memory trace*, entretanto o simulador usa arquivos para este fim. O conteúdo da cache de dados é um arquivo com extensão *.data*, e o de instruções um arquivo com extensão *.text*.

Este simulador possui uma interface gráfica com dois ambientes: a primeira é destinada à demonstração da tecnologia de cache *loop-way* e a comparação de desempenho entre a cache com e sem a tecnologia. O segundo ambiente é o modo de ensino, que representa a parte didática deste simulador destinada à demonstração do funcionamento da memória cache.

Ao acessar o ambiente de ensino, a janela de configuração é mostrada. O usuário escolhe primeiramente se deseja ou não separar a cache de dados e instruções, depois seleciona a política de escrita e atualização. Já a arquitetura de cache é determinada pelos campos: *Sets*; *Ways* e *Words/Linha*. O tipo e a capacidade são automaticamente calculados pelo programa.

Após a configuração, a tela de simulação é mostrada. Ao lado esquerdo estão as informações da memória RAM divididas em duas áreas: área de dados e área de instruções. No lado direito da janela ficam as informações do processador, que informam a quantidade de instruções que foram requisitadas, bem como o tipo da instrução atual e o seu endereço.

No lado inferior, existe um campo de desempenho, que apresenta um resumo do desempenho geral do sistema, sendo possível consultar também o desempenho das duas caches separadamente. Na parte central temos as informações da memória cache. A simulação pode acontecer passo a passo, onde podem ser observadas alterações por animações na interface, ou diretamente.

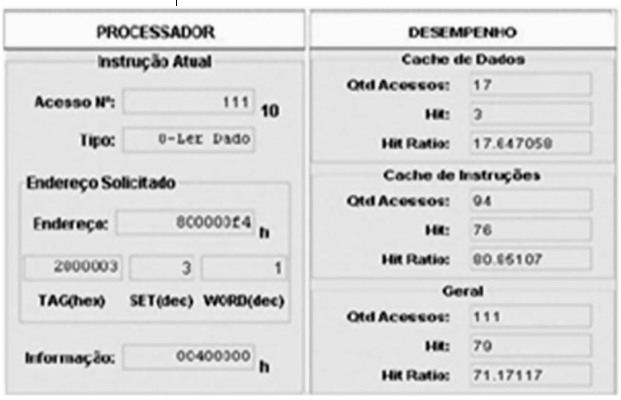

Figura 9. Algumas informações exibidas em uma simulação do LBGCache.

3.8 SMPCache

Este simulador foi desenvolvido no Departamento de Informática da Universidade de Extremadura, na Espanha. Este simulador possui a ideia principal de avaliar o desempenho da memória cache em ambientes multiprocessados, onde temos múltiplos processadores e apenas uma memória compartilhada entre eles (*Symmetric Multi-Processing Cache*), mas também possibilita simulações de arquiteturas simples.

A ferramenta implementa as arquiteturas de cache: mapeamento direto; associativo e associativo por conjunto. Políticas de substituição: FIFO; LRU; LFU e aleatória e política de escrita: *write back*. Outros parâmetros foram implementados, como a quantidade de processadores: de 1 a 8; A escolha do protocolo de coerência: *Modified Shared Invalid* (MSI). *Modified Exclusive Shared Invalid* (MESI), *Dragon*; Tamanho de bits da palavra: 8 a 64. Este simulador não apresenta simulação de cache multi-nível.

O *memory trace* pode ser obtido tanto no site do simulador, quanto feito pelo usuário seguindo o formato: <operação> <endereço>. O simulador dispõe também de uma ferramenta chamada *TraceConv*, que realiza a conversão dos formatos de *Limes* e canônico para o formato de *trace*.

A configuração da simulação é feita na janela principal do programa. Depois de selecionada a arquitetura, o ambiente de simulação é apresentado. Na parte superior da janela são exibidas as informações dos processadores e de cada cache, que estão conectados por um barramento compartilhado com a memória principal.

Na parte inferior esquerda, existem os contadores que exibem as transferências no barramento, as transferências de blocos de memória, e as transições dos estados. Logo após, é possível encontrar as informações do protocolo de coerência. Na parte central do software é exibido o progresso da simulação e uma tabela com a quantidade de acessos a memória, número de instruções realizadas, quantidade dados que foram lidos ou escritos, as taxas gerais de acertos e erros decorrentes da simulação.

Na parte inferior direita, existem os botões de controle da simulação, que pode ser configurada para ocorrer passo a passo ou de forma direta. Após o término da simulação a janela de estatísticas é mostrada, na qual as informações mais importantes são mostradas através de gráficos.

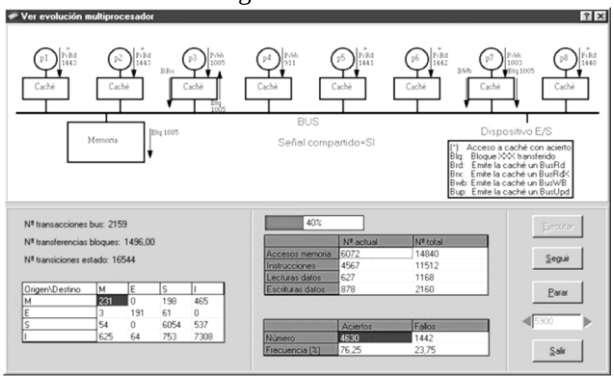

Figura 10. Tela do SMPCache.

4. Comparação entre os simuladores

Essa seção apresenta a comparação entre os simuladores selecionados, através do estudo de seus respectivos artigos e outras documentações encontradas.

4.1 Parâmetros de comparação

Após a análise dos simuladores, algumas características fundamentais foram discutidas e essa seção mostra os requisitos essenciais desejados em um simulador.

Como a interface gráfica é a forma de interação com o usuário, ela deve ser intuitiva, de forma que a ferramenta seja fácil de utilizar. É importante que o processo de utilização do ambiente de simulação aconteça em etapas. O usuário deve passar por cada etapa, inserindo as informações solicitadas sobre a arquitetura simulada até a conclusão da simulação e exibição dos resultados.

A visibilidade das informações é fundamental, a ferramenta deve mostrar como é feito o acesso à memória cache, através da exibição de todos os parâmetros envolvidos no controle da mesma. Ao final da simulação é esperado um relatório, cujo conteúdo mostraria uma descrição detalhada do desempenho do sistema.

Para evitar erros, a geração do *memory trace* deve seguir um modelo fácil e deve ser feita de forma manual ou aleatória dentro do aplicativo.

É essencial a disponibilização do aplicativo e de seu código-fonte. Alguns dos trabalhos desenvolvidos não estão disponíveis para *download* e o contato dos desenvolvedores não é mais válido. Após o desenvolvimento da ferramenta, esta deve ser disponibilizada de forma simples, através, por exemplo, de sites que ofereçam hospedagem permanente.

4.2 Estudo comparativo

Após a análise dos artigos dos simuladores de acordo com os critérios escolhidos, foi possível verificar que a maioria dos simuladores analisados não atende a alguns critérios.

Dos simuladores selecionados, o KSH, DCMSim, Cachesim e o SMPCache não apresentam a opção de uma cache dividida para dados e instruções, enquanto que todos os outros possuem tal recurso.

Em relação à quantidade de níveis de memória cache, apenas o Dinero IV e o MSCSim apresentam a possibilidade de realizar simulações com vários níveis de cache.

Os tipos de acesso sequencial e paralelo, bem como o tempo de cada acesso são características apenas do KSH e do MSCSim, os outros simuladores não disponibilizam tal opção.

Os erros de compulsório, conflito e capacidade que podem ocorrer em um acesso à memória cache são pertinentes apenas aos simuladores Dinero IV e MSCSim.

Em relação ao *memory trace*, a maioria dos simuladores usa arquivos de texto elaborados pelo usuário, baseados em um modelo pré-definido. Embora não seja uma tarefa tão complexa, uma função randômica que gere o *memory trace* é importante, sendo o MSCSim o único simulador que possui tal recurso. O Dinero IV e o LBGCache utiliza um *trace* gerado pelo próprio programa.

No quesito intuitividade, os que mais demonstraram deficiência foram o Dinero III e o Dinero IV, pois não apresentam interface gráfica e nem comandos simples para as operações, que são exclusivamente via console. Bem como o Cachesim, que apresenta poucos recursos visuais.

Todos os outros possuem interfaces gráficas, variando em complexidade. A ferramenta que se mostrou mais completa foi o MSCSim, apresentando uma interface bem estruturada.

Na análise da visibilidade de informações, o MSCSim se destaca, exibindo uma gama de informações em sua simulação, além de possuir animações que ajudam a visualizar as modificações. O DCMSim e o KSH apresentam apenas as informações básicas, enquanto que o Dinero III e o Dinero IV não apresentam informações durante sua simulação, apresentando apenas um arquivo de relatório ao término. Os outros simuladores apresentam apenas informações básicas da simulação. O SMPCache se diferencia por mostrar a evolução da memória cache através de um formato gráfico.

A Tabela 1 mostra as características básicas, que foram os parâmetros de comparação existentes nos simuladores, e outras características adicionais que se mostraram presentes nos mesmos.

A maioria dos simuladores apresentaram problemas de disponibilidade, tanto do próprio aplicativo, quanto do seu respectivo código-fonte. O Dinero IV é disponível no site do autor. Para a obtenção do MSCSim, faz-se necessário o envio de um e-mail de requisição para o contato disponibilizado no site do simulador.

4.3 Resultados

A partir do estudo comparativo realizado, podemos verificar que há uma grande carência nas ferramentas de simulação. Mesmo havendo vários trabalhos publicados, não é possível encontrar facilmente os simuladores, exceto o Dinero IV, tanto por estes não estarem mais disponibilizados, quanto pela impossibilidade de contato com os idealizadores.

Para a utilização do SMPCache, é necessário o preenchimento de um formulário por um professor da instituição de ensino que deve ser assinado pela instituição e enviado por correio ou fax para um *e-mail* fornecido no site.

Dos trabalhos selecionados, o MSCSim se destaca, sendo a ferramenta mais completa em todos os quesitos avaliados em nossa análise. Ele possui a interface com mais informações e as animações permitem uma melhor visualização das mudanças durante a simulação. Já o Dinero III obteve o pior desempenho. A operação via console e a falta de informações durante a simulação, tornam o uso e a compreensão difíceis, sendo não indicado para o ensino.

5. Conclusões

Através do estudo comparativo realizado entre os nove simuladores, conclui-se que o MSCSim é a ferramenta que atende as principais características que um simulador didático deve possuir.

Portanto, foi percebida a carência da existência de simuladores que possam auxiliar o ensino sobre memória cache. Assim, surge a necessidade de que sejam elaborados novos simuladores de acordo com os estudos já realizados, onde sejam implementadas as funcionalidades expostas nesse artigo.

As dificuldades encontradas não se resumem à apenas a disponibilidade de obtenção de tais

Tabela 1. Características gerais pertinentes aos simuladores

	SIMULADORES							
CARACTERÍSTICAS	KSH	Dinero III	DineroIV	MSCSim	DCMSim	Cachesim	LBGCache	SMPCache
Simulação de cache dividida	NÃO	SIM	SIM	SIM	NÃO	NÃO	SIM	NÃO
Simulação de multi-níveis	NÃO	NÃO	SIM	SIM	NÃO	NÃO	NÃO	NÃO
Tipo de acesso (seq. e paral.)	SIM	NÃO	NÃO	SIM	NÃO	NÃO	NÃO	NÃO
Políticas de substituição	FIFO/LRU	FIFO/LRU/ Aleatória	FIFO/LRU/ Aleatória	FIFO/LRU	FIFO/LRU	FIFO/LRU/ Aleatória	FIFO/LRU/ Aleatória	FIFO/LRU/ Aleatória
Tempo de acesso	SIM	NÃO	NÃO	SIM	NÃO	NÃO	NÃO	NÃO
Compulsório/Conflito/ Capacidade	NÃO	NÃO	SIM	SIM	NÃO	NÃO	NÃO	NÃO
Gerador de memory trace	NÃO	NÃO	NÃO	SIM	NÃO	NÃO	NÃO	NÃO
Apresentação de estatísticas	SIM	NÃO	NÃO	SIM	SIM	SIM	SIM	SIM
Interface didática	SIM	NÃO	NÃO	SIM	SIM	NÃO	SIM	SIM
Animação	NÃO	NÃO	NÃO	SIM	NÃO	NÃO	SIM	SIM
Disponível para download	NÃO	NÃO	SIM	NÃO	SIM	NÃO	NÃO	NÃO
Código-fonte disponível	NÃO	NÃO	SIM	NÃO	NÃO	NÃO	NÃO	NÃO

aplicativos, mas também devido à inexistência de recursos que melhorem a didática das aulas de Arquitetura de Computadores e este fato restringem tais aplicativos a funcionarem apenas como programas de avaliação de desempenho.

As novas ferramentas devem, primeiramente, ser disponibilizadas de forma simples, através, por exemplo, de um serviço de hospedagem permanente de arquivos. O projeto da interface gráfica deve primar por criar um ambiente agradável, no qual o aluno possa obter as informações do funcionamento e de desempenho de várias organizações diferentes de memória cache.

A opção de gerar relatórios ao final de cada simulação é essencial para a comparação entre os sistemas que empregam memória cache e os que não empregam e para mensurar o desempenho que pode ser obtido com a divisão da cache em diferentes níveis, bem como as outras vantagens proporcionadas por essas memórias. Recursos automatizados para geração de *memory trace* ou até mesmo para análise de métricas temporais também são fundamentais nesse ambiente.

As considerações realizadas neste trabalho podem servir como referência para a elaboração de novos simuladores, indicando quais os recursos primordiais que os mesmos necessitam possuir, e reflete a principal contribuição deste artigo.

Em conjunto com os artigos analisados nesse estudo, este trabalho complementa as pesquisas realizadas sobre simuladores de memória cache mostrando um estudo comparativo, de modo a facilitar a escolha ou elaboração de uma ferramenta de simulação que possa ser empregada na disciplina de Arquitetura de Computadores.

6. Referências

[1] Coutinho, L. M. N.; Mendes, J. L. D.; Martins, C. A. P. S. “MSCSim – Simulador de Memória Cache, Split e Multiníveis”, VI Workshop de Sistemas Computacionais de Alto Desempenho (WSCAD), 2005, pp. 193-196.

[2] Djordjevic, J.; Nikolic, B.; Mitrovic, M. “A Memory System for Education”, The Computer Journal, Vol. 48, No. 6, 2005, pp. 630-641.

[3] Yurcik, W.; Wolffe, G. S.; Holliday, M. A; “A Survey of Simulators Used in Computer Organization/Architecture Courses”, Summer Computer Simulation Conference (SCSC), Society for Computer Simulation (SCS), Orlando, FL, EUA, 2001.

[4] Ribeiro, A. S.; Duarte, R. P. “KSH: Simulador de memória cache com carregamento dinâmico de módulos e execução de script de configuração”, IV Workshop em Sistemas Computacionais de Alto Desempenho (WSCAD), 2003, pp. 160-163.

[5] Cordeiro, E. S.; Stefani, I. G. A.; Soares, T.; Martins, C. A. P. S. “DCMSim: Didactic Cache Memory Simulator”, Frontiers in Education Conference – FIE 2003, Boulder – Colorado, v. 2003, pp. F1C14-F1C19, 2003. PDF.

[6] Hill, M. D., J. Dinero III. Disponível em: http://heather.cs.ucdavis.edu/~matloff/Dinero/Man.html.

[7] Hill, M. D., J. Dinero IV Trace-Driven Uniprocessor Cache Simulator. Disponível em: http://www.cs.wisc.edu/~markhill/DineroIV/.

[8] Prete, C. A. “Cachesim: A Graphical Software Environment to Support the Teaching of Computer Systems with Cache Memories, Proceedings of the 7th SEI CSEE Conference on Software Engineering Education, pp. 317-327, 1994.

[9] Moreira K.; Pontes, B.; Fernandes, G.; Vidal, J. M.; Wanderley Netto, E. B. “LBGCache V2.0: Simulador Didático para Análise do Comportamento e Funcionamento Estrutural de Memórias Caches”. In: Anais do III Congresso de Iniciação Científica do CEFETRN. Natal, Brasil, 2005.

[10] Vega, M. A.; Martin, R.; Zarallo, F.A.; Sánchez J. M.; Gómez, J. A. "SMPCache: Simulador de Sistemas de Memoria Caché en Multiprocesadores Simétricos". In: Actas de las Xi Jornadas de Paralelismo, 2000, pp.3-8.

[11] Rodrigues, J. C.; Xavier, M. A. S.; Lima Júnior, O. A.; “Simuladores de Memória Cache, um Estudo Comparativo Direcionado ao Ensino”. In: Anais do VI Workshop sobre Educação em Arquitetura de Computadores (WEAC), 2011, pp. 7-12.

[12] Hennessy, J. L.; Patterson, D. A. “Computer Organization and Design: The Hardware/Software Interface”, 3rd Edition, Morgan Kaufman, 2005.

[13] William Stallings. “Organização e Arquitetura de Computadores”, 5ª Edição, Pearson Prentice Hall, 2002.

[14] Prima Cache Simulator. Disponível em: http://www.dsi.unimo.it/staff/st36/imagelab/prima.html.

[15] Moreira, L.; Pontes, B.; Fernandes, G.; Vidal, J. M.; Wanderley Netto, E. B.; Loop-way Cache. In Anais do Workshop em Sistemas Computacionais de Alto Desempenho, Oct, 24-27, pp. 197-200 Rio de Janeiro, Brazil, 2005.