Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

SCC0112 – Organização de Computadores Digitais I

# 2º Trabalho Prático:

# Análise de Desempenho de Memória Cache

Carlos Humberto dos Santos Baqueta – 7987456
Elias Italiano Rodrigues – 7987251
Ricardo Issamu Fukuda Gunzi – 7986729

São Carlos, 3 de dezembro de 2015

# Sumário

**1 Introdução** **2**

**2 Desenvolvimento** **2**

2.1 O simulador Amnesia . . . . . 2

2.2 Arquivos de *trace* . . . . . 3

2.2.1 Não-espacial e não-temporal . . . . . 3

2.2.2 Espacial . . . . . 4

2.2.3 Temporal . . . . . 4

2.2.4 Espacial e temporal . . . . . 5

2.2.5 Aleatório . . . . . 5

2.3 Arquiteturas . . . . . 6

2.3.1 Base . . . . . 6

2.3.2 Variadas . . . . . 6

2.4 Resultados das execuções . . . . . 8

2.4.1 Relação da taxa de acerto com o tamanho da *cache* . . . . . 8

2.4.2 Relação da taxa de acerto com o tamanho do bloco . . . . . 9

2.4.3 Relação da taxa de acerto com o nível de associatividade . . . . . 9

2.4.4 Relação da taxa de acerto com o algoritmo de substituição . . . . . 10

2.4.5 Relação da taxa de acerto com o número de *caches* . . . . . 10

2.4.6 Combinação das arquiteturas . . . . . 10

**3 Conclusão** **11**

**Referências** **11**

# 1 Introdução

Este segundo trabalho da disciplina de Organização de Computadores Digitais I tem como objetivo analisar o desempenho de uma hierarquia de memórias *cache* com relação à **taxa de acerto** (*hit rate*). Os resultados para a análise foram gerados por meio de simulações em *software*. Para isso, foi escolhido o simulador Amnesia [1] e nele foram executados diferentes *traces* sobre uma variedade de arquiteturas de memórias *cache* definidas de acordo com as limitações do simulador.

O desenvolvimento deste trabalho está organizado da seguinte maneira: primeiramente é apresentado o simulador Amnesia, em seguida são comentados e mostrados os arquivos de *trace* e as arquiteturas definidas pelo grupo do trabalho e então, por meio de tabelas e gráficos, são analisadas as taxas de acerto com relação as características das arquiteturas (tamanho da *cache*, tamanho do bloco, associatividade, algoritmo de substituição e quantidade de *caches*).

Os arquivos utilizados neste trabalho devem acompanhar este documento.

# 2 Desenvolvimento

## 2.1 O simulador Amnesia

De acordo com a página oficial do Amnesia [1]:

> "O Amnesia é um simulador de hierarquia de memória, de sistemas computacionais, com fins didáticos. Ele permite simular o comportamento de registradores em um processador, memórias cache, memória principal e memória virtual paginada.
>
> O Amnesia 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 usada. Diferentes configurações da hierarquia de memória podem ser estabelecidas e comparadas durante as atividades de simulação."

Isso faz desse simulador uma boa escolha para este trabalho, pois atende à necessidade de definir diferentes configurações de hierarquia de memória (arquiteturas) para as simulações. E ainda, é um *software* desenvolvido e mantido pela própria instituição de ensino da disciplina deste trabalho:

> "O simulador está em desenvolvimento, desde 2007, por alunos de graduação e da pós-graduação do Instituto de Ciências Matemáticas e de Computação (ICMC) da Universidade de São Paulo (USP), orientados pelos professores Paulo Sérgio Lopes de Souza e Sarita Mazzini Bruschi."

**Observação**: importante ressaltar que o Amnesia é um simulador com propósito didático que incorpora características de Objetos de Aprendizagem (OA) e Recursos Educacionais Abertos (REA) para auxiliar professores e alunos durante o processo de aprendizagem. Portanto, não é adequado usá-lo para um processo de *benchmark* que também não é o propósito deste trabalho.

## 2.2 Arquivos de *trace*

Arquivos de *trace*, ou arquivos de rastro, são arquivos de texto ASCII que contém em cada linha uma dupla: rótulo (decimal) e endereço (hexadecimal). Qualquer outra informação é vista como um comentário. O rótulo representa uma operação de memória a ser feita sobre o endereço. O rótulo pode ser:

- 0: leitura de dados;
- 1: gravação de dados;
- 2: busca de instrução;
- 3: registro escape (tratado como tipo de acesso desconhecido);
- 4: registro escape (operação de cache flush).

**Observação**: como o objetivo deste trabalho é analisar a taxa de acerto e não foi considerada uma política de escrita para as *caches*, então o rótulo utilizado para as operações é indiferente. Por definição, foi utilizado o **rótulo 2** para todas operações dos arquivos de *trace*.

Os arquivos de *trace* para este trabalho foram criados de tal modo que as heurísticas do princípio de localidade espacial e temporal fossem exploradas. Assim, criou-se quatro arquivos tais que: um não houvesse localidade espacial e nem temporal, um houvesse somente localidade espacial, um houvesse somente localidade temporal e outro com ambos os princípios de localidade. Além disso, criou-se um quinto arquivo de *trace* com endereços aleatórios na tentativa de avaliar o algoritmo de substituição.

Para gerar os arquivos de *trace*, usou-se um programa simples em linguagem C, `trace_gen.c`, que imprime na saída padrão um arquivo de *trace* com um dos casos citados acima.

Com exceção do primeiro *trace*, a **quantidade de operações** foi definida em **32768**, pois assim é possível que um *trace* acesse cada palavra da memória principal cujo tamanho está especificado mais adiante na Seção 2.3.

Devido ao tamanho dos arquivos de *trace*, não é conveniente inseri-los inteiramente neste documento. Então seguem trechos dos arquivos com comentários sobre suas características.

### 2.2.1 Não-espacial e não-temporal

Gerado com:

```
for (i = 0; i < 2978; i++) {
    printf("2 %x\n", (i * 11));
}
```

Trecho do arquivo `trace-00.txt`:

```
2 0
2 b
2 16
2 21
2 2c
2 37
...
```

Neste *trace* os endereços das palavras tem um intervalo suficientemente grande: 11 que é maior do que qualquer uma das quantidades de palavras por bloco presentes nas arquiteturas definidas. Isso proporciona característica não-espacial. Além disso, não é feito acesso repetido a um mesmo endereço o que proporciona característica não-temporal. Em particular, a quantidade de operações deste *trace* é menor, pois caso contrário seriam gerados endereços de palavras inexistentes na memória principal.

### 2.2.2 Espacial

Gerado com:

```
for (i = 0; i < 32768; i++) {
    printf("2 %x\n", i);
}
```

Trecho do arquivo `trace-01.txt`:

```
2 0
2 1
2 2
2 3
2 4
2 5
2 6
...
```

Neste *trace* os endereços das palavras são bem próximos (consecutivos), porém com nenhum acesso repetido. Isso proporciona característica espacial e não-temporal.

### 2.2.3 Temporal

Gerado com:

```
int *address = (int *)malloc(sizeof(int) * 96);

address[0] = 0;
for (i = 1; i < 96; i++) {
    address[i] = address[i-1] + 11;
}
for (i = 0; i < 32768; i++) {
    printf("2 %x\n", address[i % 96]);
}

free(address);
```

Trecho do arquivo `trace-10.txt`:

```
2 0
2 b
2 16
2 21
2 2c
...
2 0  nonagésima sétima operação: começa a repetir
2 b
2 16
...
```

Neste *trace* os endereços são de palavras distantes (novamente 11 posições de distância que é suficiente para não serem do mesmo bloco em nenhuma das *caches*), porém a cada 96 operações são realizados acessos repetidos aos mesmos endereços anteriores. Isso proporciona característica temporal e não-espacial.

### 2.2.4 Espacial e temporal

Gerado com:

```
for (i = 0; i < 32768; i++) {
    printf("2 %x\n", i % 320);
}
```

Trecho do arquivo `trace-11.txt`:

```
2 0
2 1
2 2
2 3
2 4
2 5
...
2 0 tricentésima vigésima primeira operação: começa a repetir
2 1
2 2
...
```

Neste *trace* os endereços das palavras são bem próximos (consecutivos) e ocorre repetição dos endereços a cada 320 operações. Isso proporciona característica espacial e temporal.

### 2.2.5 Aleatório

Gerado com:

```
srand(time(NULL));
for (i = 0; i < 32768; i++) {
    printf("2 %x\n", rand() % 192);
}
```

Trecho do arquivo `trace-rand.txt`:

```
2 16
2 2a
2 59
2 b9
2 40
2 65
...
```

Neste *trace* os endereços das palavras foram gerados aleatoriamente dentro um intervalo de 0 a 191 usando a função `rand()` da linguagem C.

## 2.3 Arquiteturas

Foram definidas uma arquitetura base e outras dez arquiteturas com configurações variadas a partir da arquitetura base. Toda **palavra** tem tamanho fixo de **4 bytes (32 bits)** e a **memória principal é endereçada a byte**.

Como citado anteriormente, o propósito deste trabalho e do simulador Amnesia não é o de *benchmark*, portanto as configurações foram escolhidas de acordo com as limitações do simulador e também dos computadores pessoais do grupo do trabalho usados para executar o simulador. O tamanho da **memória principal** é fixo de **128KiB** (portanto 32768 palavras) e os tamanhos para as ***caches*** podem ser de **512B**, **1KiB** e **2KiB**.

No Amnesia, o arquivo de arquitetura é definido no formato XML e um exemplo de arquitetura pode ser consultado no manual que acompanha o simulador.

**Observação**: as *caches* definidas para este trabalho são todas do tipo **unified**, ou seja, não há separação dentro da *cache* entre as palavras que representam instruções e as que representam dados como há no tipo *split*. Como políticas de escrita não foram consideradas e por definição as operações de memória são apenas de busca de instrução (rótulo 2), não há motivo para escolher o tipo *split*.

### 2.3.1 Base

A arquitetura base foi definida no arquivo `arch-00.xml` e possui as configurações especificadas na Tabela 1.

Tabela 1: Configuração da arquitetura base.

| **arch-00** | | | | |
|---|---|---|---|---|
| **Cache** | **Tamanho** | **Palavras por bloco** | **Mapeamento** | **Substituição** |
| **L1** | 128 palavras (512B) | 2 | Direto | — |

### 2.3.2 Variadas

As dez arquiteturas variadas foram definidas nos respectivos arquivos `arch-01.xml`, `arch-02.xml`, `...`, `arch-10.xml` e possuem as configurações especificadas na Tabela 2.

Tabela 2: Configurações variadas. Em itálico estão os valores alterados com relação a arquitetura base.

| **arch-01** | | | | |
|---|---|---|---|---|
| **Cache** | ***Tamanho*** | **Palavras por bloco** | **Mapeamento** | **Substituição** |
| **L1** | *256 palavras (1KiB)* | 2 | Direto | — |

| **arch-02** | | | | |
|---|---|---|---|---|
| **Cache** | ***Tamanho*** | **Palavras por bloco** | **Mapeamento** | **Substituição** |
| **L1** | *512 palavras (2KiB)* | 2 | Direto | — |

| **arch-03** | | | | |
|---|---|---|---|---|
| **Cache** | **Tamanho** | ***Palavras por bloco*** | **Mapeamento** | **Substituição** |
| **L1** | 128 palavras (512B) | *4* | Direto | — |

| **arch-04** | | | | |
|---|---|---|---|---|
| **Cache** | **Tamanho** | ***Palavras por bloco*** | **Mapeamento** | **Substituição** |
| **L1** | 128 palavras (512B) | *8* | Direto | — |

| **arch-05** | | | | |
|---|---|---|---|---|
| **Cache** | **Tamanho** | **Palavras por bloco** | ***Mapeamento*** | **Substituição** |
| **L1** | 128 palavras (512B) | 2 | *Associativo (2)* | FIFO |

| **arch-06** | | | | |
|---|---|---|---|---|
| **Cache** | **Tamanho** | **Palavras por bloco** | ***Mapeamento*** | **Substituição** |
| **L1** | 128 palavras (512B) | 2 | *Associativo (4)* | FIFO |

| **arch-07** | | | | |
|---|---|---|---|---|
| **Cache** | **Tamanho** | **Palavras por bloco** | **Mapeamento** | ***Substituição*** |
| **L1** | 128 palavras (512B) | 2 | *Associativo (4)* | *LRU* |

| **arch-08** | | | | |
|---|---|---|---|---|
| ***Cache*** | **Tamanho** | **Palavras por bloco** | **Mapeamento** | **Substituição** |
| **L1** | 128 palavras (512B) | 2 | Direto | — |
| ***L2*** | *256 palavras (1KiB)* | 2 | Direto | — |

| **arch-09** | | | | |
|---|---|---|---|---|
| ***Cache*** | **Tamanho** | **Palavras por bloco** | **Mapeamento** | **Substituição** |
| **L1** | 128 palavras (512B) | 2 | Direto | — |
| ***L2*** | *256 palavras (1KiB)* | 2 | Direto | — |
| ***L3*** | *512 palavras (2KiB)* | 2 | Direto | — |

| **arch-10** | | | | |
|---|---|---|---|---|
| ***Cache*** | **Tamanho** | **Palavras por bloco** | **Mapeamento** | **Substituição** |
| **L1** | 128 palavras (512B) | 2 | Direto | — |
| ***L2*** | *256 palavras (1KiB)* | *4* | *Associativo (2)* | *LRU* |
| ***L3*** | *512 palavras (2KiB)* | *8* | *Associativo (4)* | *LRU* |

## 2.4 Resultados das execuções

Para cada arquitetura, foram executados os cinco arquivos de *trace* no simulador Amnesia. As taxas de acerto coletadas são mostradas na Tabela 3.

Tabela 3: Taxa de acerto para cada arquitetura e arquivo de *trace*.

| arch- | Cache | trace-00 | trace-01 | trace-10 | trace-11 | trace-rand | Média |
|---|---|---|---|---|---|---|---|
| **00** | L1 | 0 | 0.5 | 0.332489 | 0.5 | 0.66537476 | 0.49946594 |
| **01** | L1 | 0 | 0.5 | 0.97628784 | 0.796875 | 0.9970703 | 0.817558285 |
| **02** | L1 | 0 | 0.5 | 0.97628784 | 0.9951172 | 0.9970703 | 0.867118835 |
| **03** | L1 | 0 | 0.75 | 0 | 0.75 | 0.66882324 | 0.54220581 |
| **04** | L1 | 0 | 0.875 | 0 | 0.875 | 0.6715698 | 0.60539245 |
| **05** | L1 | 0 | 0.5 | 0.020751953 | 0.5 | 0.6673279 | 0.42201996325 |
| **06** | L1 | 0 | 0.5 | 0 | 0.5 | 0.66674805 | 0.4166870125 |
| **07** | L1 | 0 | 0.5 | 0 | 0.5 | 0.66656494 | 0.416641235 |
| **08** | L1 | 0 | 0.5 | 0.332489 | 0.5 | 0.66537476 | 0.49946594 |
| | L2 | 0 | 0 | 0.96447676 | 0.59375 | 0.99124485 | 0.6373679025 |
| **09** | L1 | 0 | 0.5 | 0.332489 | 0.5 | 0.66537476 | 0.49946594 |
| | L2 | 0 | 0 | 0.96447676 | 0.59375 | 0.99124485 | 0.6373679025 |
| | L3 | 0 | 0 | 0 | 0.97596157 | 0 | 0.2439903925 |
| **10** | L1 | 0 | 0.5 | 0.332489 | 0.5 | 0.66537476 | 0.49946594 |
| | L2 | 0 | 0.5 | 0.86983955 | 0.6982422 | 0.99562246 | 0.7659260525 |
| | L3 | 0 | 0.5 | 0.95539165 | 0.9919094 | 0.5 | 0.7368252625 |

### 2.4.1 Relação da taxa de acerto com o tamanho da *cache*

Para avaliar o impacto do tamanho da *cache* na taxa de acerto, contrastou-se as taxas de acerto das arquiteturas arch-00, arch-01 e arch-02. O resultado pode ser visto na Figura 1.

Foi possível observar que o aumento no tamanho da *cache* implicou no aumento da taxa de acerto como esperado. Quanto maior o tamanho da *cache*, menor é a quantidade de falhas por capacidade, pois menos blocos disputam por uma mesma posição.

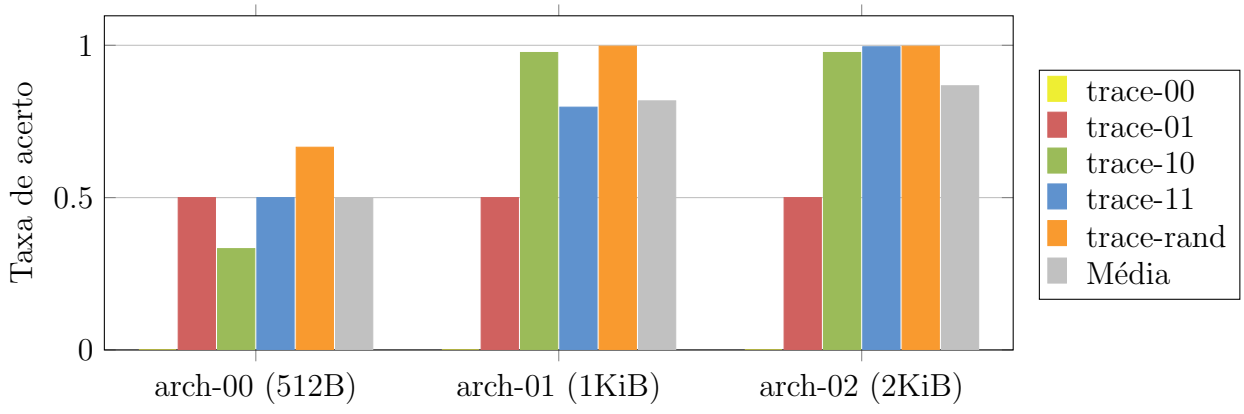


Figura 1: Comparação da taxa acerto com relação ao tamanho da *cache*.

### 2.4.2 Relação da taxa de acerto com o tamanho do bloco

Para avaliar o impacto do tamanho do bloco na taxa de acerto, contrastou-se as taxas de acerto das arquiteturas arch-00, arch-03 e arch-04. O resultado pode ser visto na Figura 2.

Quanto ao trace-rand, houve pouca variação na taxa de acerto, mas foi possível observar que a média de acertos aumentou conforme o tamanho do bloco.

Significativamente, observou-se aumento para os trace-01 e trace-11 e declínio para zero do trace-10. Isso aconteceu pois, uma vez que o tamanho do bloco foi aumentado, os trace-01 e trace-11 que possuem princípio de localidade espacial tem mais chance de acerto já que uma maior quantidade de palavras são copiadas para a *cache*. Quanto ao declínio para zero do trace-10 deve-se ao fato dele possuir somente característica temporal, o que leva a um pior desempenho com blocos maiores, pois ocorre sobrescrita das palavras que possivelmente seriam utilizadas novamente durante a execução.

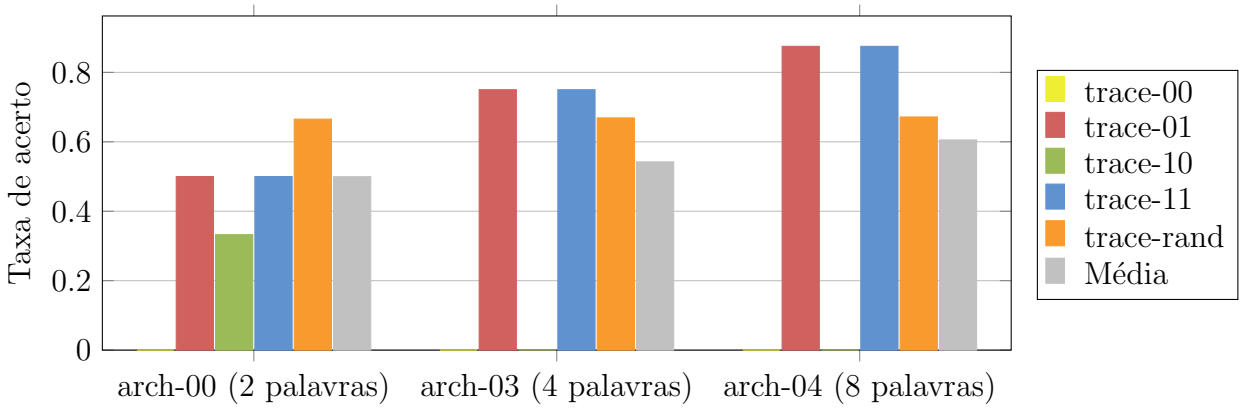


Figura 2: Comparação da taxa acerto com relação ao tamanho do bloco.

### 2.4.3 Relação da taxa de acerto com o nível de associatividade

Para avaliar o impacto do nível de associatividade na taxa de acerto, contrastou-se as taxas de acerto das arquiteturas arch-00, arch-05 e arch-06. O resultado pode ser visto na Figura 3.

Quanto ao trace-01 e trace-11, que possuem característica espacial, não houve diferença na taxa de acerto, pois o tamanho do bloco não foi aumentado. Por outro lado, para o trace-10 que possui somente característica temporal, houve diminuição na taxa de acerto conforme o aumento do nível de associatividade. Esse comportamento é esperado para este *trace*, pois foi aumentado o nível de associatividade sem aumentar o tamanho do bloco o que gerou mais falhas para localidade temporal.

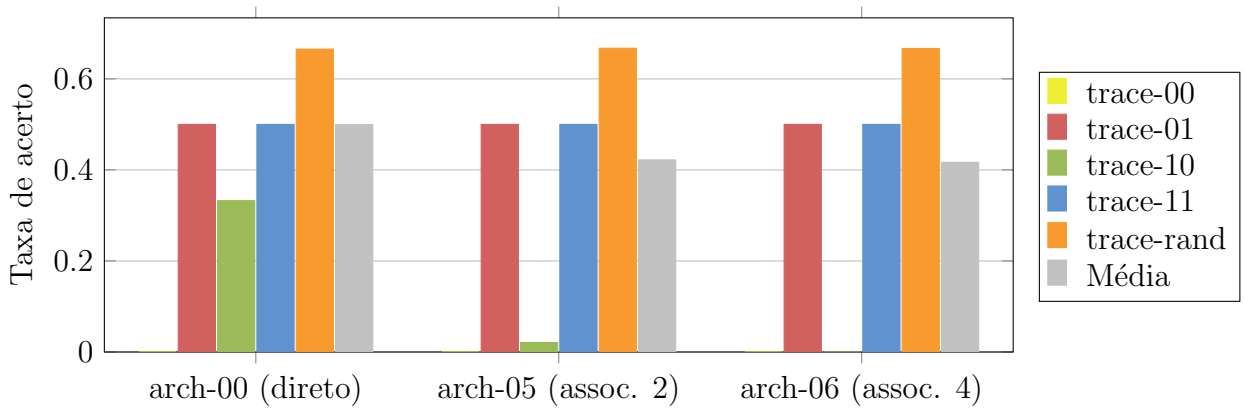


Figura 3: Comparação da taxa acerto com relação o nível de associatividade.

### 2.4.4 Relação da taxa de acerto com o algoritmo de substituição

Para avaliar a taxa de acerto com relação ao algoritmo de substituição, observou-se as arquiteturas arch-06 e arch-07. Pela Tabela 3, pode-se notar que a diferença entre as duas foi mínima, havendo pequena alteração somente para o trace-rand. Isso mostra que os arquivos de *trace* gerados não foram suficientes para avaliar o algoritmo de substituição. Com exceção do trace-rand, os demais *traces* tem padrões de acesso sistemáticos o que implica na indiferença do algoritmo de substituição escolhido.

### 2.4.5 Relação da taxa de acerto com o número de *caches*

Para avaliar o impacto do número de *caches* na taxa de acerto, contrastou-se as taxas de acerto das arquiteturas arch-00, arch-08 e arch-09. Notoriamente o aumento do número de *caches* diminui a quantidade de acessos à memória principal, pois proporciona mais alternativas de acesso. O problema que o número de *caches* pode ocasionar é com relação à atualização dos dados (escrita) que não foi abordada nesse trabalho.

### 2.4.6 Combinação das arquiteturas

A arquitetura arch-10 proposta é uma combinação das demais com o objetivo de conseguir um melhor desempenho. A partir da Tabela 3 constatou-se que houve um melhor desempenho, pois com exceção das arquiteturas arch-01 e arch-02 que possuem tamanho da L1 maior, a arch-10 proporcionou, em média geral, uma taxa de acerto maior que todas as demais arquiteturas.

# 3 Conclusão

O trace-00, que não possui característica de localidade espacial e nem temporal, resultou em taxas de acerto zero para todas as arquiteturas utilizadas. Isso evidencia que a hierarquia de memórias *cache* é fundamentada nesse princípio e, portanto, na ausência dele a *cache* se torna um *overhead* para sistema computacional.

De modo geral, os comportamentos da taxa de acerto conforme as variações nas características da *cache* já eram os esperados, pois é parte do conteúdo da disciplina SCC0112. Portanto, este trabalho serviu para consolidar o que foi aprendido em sala de aula por meio de uma abordagem prática ao tema definindo as arquiteturas e simulando em *software*.

# Referências

[1] Amnesia: Memory Hierarchy Simulator
Disponível em <http://amnesia.lasdpc.icmc.usp.br/>