Apêndice A - Plano de aula utilizado nas aulas práticas

Importante salientar que para todas as arquiteturas, as configurações dos parâmetros *ciclesPerAccessRead, ciclesPerAccessWrite e timeCicle* serão constantes conforme a seguir:

|  |  |  |  |
| --- | --- | --- | --- |
| **Memória / Configuração** | ***ciclesPerAccessRead*** | ***ciclesPerAccessWrite*** | ***timeCicle*** |
| ***Cache*** | 1 | 2 | **1** |
| **Principal** | 1 | 2 | **10** |

1. Ausência de memória *cache* entre processador e memória principal.

**Motivação:** Demonstrar aos alunos que o tempo de acesso aos dados e instruções na memória principal é alto. Introduzir os benefícios obtidos com o uso de uma memória *cache*.

**Objetivo:** Destacar o tempo total de execução de uma aplicação (através do seu *trace*) sem o uso da *cache*.

**Detalhamento:** Utilizar um arquivo de arquitetura sem memória *cache* e executar um arquivo de *trace* que realiza o carregamento das mesmas instruções na memória por diversas vezes consecutivas (ênfase no princípio da localidade temporal).

Arquitetura: Architecture-01-R-s8-bs1-Cno.xml

Trace: Trace01-R50.txt

Tipo da execução: Direta

Resultados que serão enfatizados:

Acessos: 50

**Tempo total: 500**

1. Incluir memória *cache* entre o processador e memória principal.

**Motivação: Mostrar aos alunos que o uso de *cache* pode diminuir os acessos à memória principal** e consequentemente o **tempo total** de execução de um determinado programa.

**Objetivo: Reduzir o tempo total** de execução de um determinado programa com o uso de uma *cache*.

**Detalhamento:** Utilizar um arquivo de arquitetura com uma memória *cache* pequena e com nível de associatividade igual a um (mapeamento direto). Executar o arquivo de *trace* do experimento anterior que realiza o carregamento das mesmas instruções na memória diversas vezes consecutivas (princípio da localidade temporal).

Arquitetura: Architecture-02-R-s8-bs1-C1-tU-mD-s4-bs1-al1-wpWT-raFifo.xml

Trace: Trace01-R50.txt

Tipo da execução: Direta

Resultados que serão enfatizados:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memória / Taxas | Acessos | Total de acertos (***hit****s*) | Total de falhas (***mis****ses*) | **Tempo total** |
| *Cache* | 50 | 46 | 4 | **50** |
| Principal | 4 |  |  | **40** |
|  |  |  |  | **90** |

**Razão: 90/500 = 0,18 (18% do tempo total sem a memória)**

Variações que podem ser realizadas:

* Inserir mais instruções no arquivo de *trace* e observar a vantagem da memória *cache*:
  + Executar o arquivo de trace Trace-02-R1000.txt
    - Realiza o carregamento das **mesmas instruções** na memória, só que por **1000 vezes** consecutivas;
    - Destacar o **tempo total,** que será **menor do que o tempo total do experimento um (próximo da metade) só que para um número bem maior de instruções**. Além disso, enfatizar que o tempo total é bem próximo do tempo total da memória cache.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memória / Taxas | Acessos | Total de acertos (***hit****s*) | Total de falhas (***mis****ses*) | **Tempo total** |
| *Cache* | 1000 | 996 | 4 | **1000** |
| Principal | 4 |  |  | **40** |
|  |  |  |  | **1040** |

**Razão: 1040 / 10000: 0,104 (10,4% do tempo total sem a memória)**

1. Algoritmo de **mapeamento ou espalhamento – Mapeamento Direto (Associatividade 1)**.

**Motivação:** Determinar o local que o bloco da memória principal ocupa na memória *cache* e como identificar que o bloco procurado pelo processador está ou na memória *cache*.

**Objetivo:** Entender como funciona o algoritmo de mapeamento direto.

**Detalhamento:** Utilizar um arquivo de arquitetura com uma memória *cache* pequena e com nível de associatividade igual a um (mapeamento direto) e executar um arquivo de *trace* pequeno que realiza o carregamento de algumas instruções.

Arquitetura: Architecture-02-R-s8-bs1-C1-tU-mD-s4-bs1-al1-wpWT-raFifo.xml

Trace: Trace03-R.txt

Tipo da execução: Passo a passo

Resultados que serão enfatizados:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memória / Taxas | Acessos | Total de acertos (***hit****s*) | Total de falhas (***mis****ses*) | **Tempo total** |
| *Cache* | 10 | 2 | **8** | **10** |
| Principal | 8 |  |  | **80** |
|  |  |  |  | **90** |

* Enfatizar que a concorrência na memória *cache* é alta para este padrão de acesso, devido à **falta de localidade espacial.** Assim, a taxa de falhas é elevada, aumentando o tempo total de execução, pois muitos acessos à memória principal são realizados.

Conceitos que serão enfatizados:

* Unidade de **transferência de dados** entre os níveis da hierarquia de memória é **bloco**;
* Como é feita a transferência do disco para a memória principal (**blocos e palavras que estão dentro do bloco e não blocos ou blocos adjacentes**);
* Como é a **ordem de leitura** dos dados do processador até a memória principal;
* Como identificar se **a palavra / bloco procurada** pelo processador está ou não na memória *cache*;
* Taxas de **acerto** (***hit***) e **falha** (***miss***);
* O número de **total de acessos** em uma **memória *cache*** é a **soma** dos **acertos** com as **falhas**;
* Definições de *Tag*, *Set* (**conjunto**), *Word-OffSet* (**Identificador da palavra**) e *Byte-Offset* e número de bits reservados para cada um;

1. Princípio da localidade espacial.

**Motivação: Diminuir a taxa de falhas e tempo total** de execução de um programa / *trace*.

**Objetivo:** Compreender o princípio da **localidade espacial** para **diminuir a taxa de falhas** e tentar **reduzir o tempo total de execução** do *trace*.

**Detalhamento:** Utilizar um arquivo de arquitetura com uma memória *cache* pequena, com nível de associatividade igual a um (mapeamento direto) e com duas palavras por bloco. Executar um arquivo de *trace* pequeno que realiza o carregamento de **diferentes instruções** que devem ser **alocadas,** em sua grande maioria**, no mesmo bloco da memória *cache***.

Arquitetura: Architecture-03-R-s8-bs2-C1-tU-mD-s4-bs2-al1-wpWT-raFifo.xml

Trace: Trace03-R.txt

Tipo da execução: Passo a passo

Resultados que serão enfatizados:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memória / Taxas | Acessos | Total de acertos (***hit****s*) | Total de falhas (***mis****ses*) | **Tempo total** |
| *Cache* | 10 | 6 | 4 | **10** |
| Principal | 4 |  |  | **40** |
|  |  |  |  | **50** |

* Enfatizar a queda do número de falhas e o aumento do número de acertos, proporcionando um tempo total de execução menor do que o obtido no experimento anterior (50 em relação à 90).

Conceitos que serão enfatizados:

* Ressaltar que o campo ***Word-OffSet*** foi utilizado com **um bit** para **identificar a palavra acessada no bloco**;

Variações que podem ser realizadas:

* Alterar o padrão de acessos do arquivo de *trace* de forma que gere mais conflitos de blocos. Desta forma, mesmo aumentando o número de palavras por bloco ainda haverá alta taxas de falhas devido à substituição motivada pelo tipo de mapeamento direto.
  + Executar o arquivo de *trace* Trace-04-R.txt de forma direta:
    - Destacar o **tempo total** e a **taxa de falhas (*miss*),** que serão **bem maiores** em relação ao experimento anterior.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memória / Taxas | Acessos | Total de acertos (***hit****s*) | Total de falhas (***mis****ses*) | **Tempo total** |
| *Cache* | 30 | 3 | 27 | **30** |
| Principal | 27 |  |  | **270** |
|  |  |  |  | **300** |

1. Demonstrar algoritmo de Mapeamento Associativo por Conjunto (Associatividade 2)

**Motivação: Diminuir a taxa de falhas e o tempo total de execução** do programa através da **diminuição dos conflitos** de blocos alocados na memória cache.

**Objetivo:** Compreender que o **aumento no grau de associatividade** proporciona um **posicionamento de blocos mais flexível**.

**Detalhamento:** Utilizar um arquivo de arquitetura com uma memória *cache* pequena, com nível de **associatividade igual a dois** (**associativa por conjunto**) e com uma palavra por bloco. Executar o arquivo de *trace* do experimento anterior para validar que a **concorrência entre os blocos** de *cache* **diminui** devido ao **aumento da associatividade** (localidade espacial).

Arquitetura: Architecture-04-R-s8-bs1-C1-tU-mA-s4-bs1-al2-wpWT-raFifo.xml

Trace: Trace04-R.txt

Tipo da execução: Direta

Resultados que serão enfatizados:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memória / Taxas | Acessos | Total de acertos (***hit****s*) | Total de falhas (***mis****ses*) | **Tempo total** |
| *Cache* | 30 | 21 | 9 | **30** |
| Principal | 9 |  |  | **90** |
|  |  |  |  | **120** |

* Enfatizar a diminuição na taxa de falhas e a diminuição do tempo total com o aumento da associatividade.

Variações que podem ser realizadas:

* Alterar o padrão de acessos do arquivo de *trace* de forma que **utilize os blocos de apenas um conjunto** da memória *cache*, por exemplo o **conjunto zero**.
  + Executar o arquivo de trace Trace-05-R.txt de forma direta:
    - Destacar o **tempo total** e a **taxa de falhas (*miss*),** que serão **bem maiores** em relação ao experimento anterior.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memória / Taxas | Acessos | Total de acertos (***hit****s*) | Total de falhas (***mis****ses*) | **Tempo total** |
| *Cache* | 30 | 4 | 26 | **30** |
| Principal | 26 |  |  | **260** |
|  |  |  |  | **290** |

1. Demonstrar algoritmo de Mapeamento Associativo por Conjunto (Associatividade 4) – Neste caso totalmente associativo, apenas um conjunto.

**Motivação: Diminuir a taxa de falhas e o tempo total de execução** do programa através da **diminuição dos conflitos** de blocos alocados na memória cache.

**Objetivo:** Compreender que o **aumento no grau de associatividade** proporciona um **posicionamento de blocos mais flexível**.

**Detalhamento:** Utilizar um arquivo de arquitetura com uma memória *cache* pequena, com nível de **associatividade igual a quatro** (neste caso **totalmente associativa**) e com uma palavra por bloco. Executar o arquivo de *trace* do experimento anterior para validar que a **concorrência entre os blocos** de *cache* **diminui** devido ao **aumento da associatividade** (localidade espacial).

Arquitetura: Architecture-05-R-s8-bs1-C1-tU-mA-s4-bs1-al4-wpWT-raFifo.xml

Trace: Trace05-R.txt

Tipo da execução: Direta

Resultados que serão enfatizados:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memória / Taxas | Acessos | Total de acertos (***hit****s*) | Total de falhas (***mis****ses*) | **Tempo total** |
| *Cache* | 30 | 26 | 4 | **30** |
| Principal | 4 |  |  | **40** |
|  |  |  |  | **70** |

* Enfatizar a diminuição na taxa de falhas e a diminuição do tempo total com o aumento da associatividade.

1. Política de substituição.

**Motivação:** Determinar qual bloco será substituído quando utiliza-se um posicionamento de blocos mais flexível através do mapeamento associativo por conjunto e totalmente associativo.

**Objetivo:** Compreender como funcionam as políticas de substituição FIFO (*First In First Out*) e LRU (*LeastRecentlyUsed*) para algoritmos de mapeamento associativo por conjunto e totalmente associativo**. Ressaltar que para mapeamento direto (associatividade um) não é necessária uma política de substituição, pois cada local da memória principal é mapeado diretamente para um local exato na *cache***.

**Detalhamento:** Utilizar um arquivo de arquitetura com uma memória *cache* pequena, com nível de **associatividade igual a dois** e com uma palavra por bloco. Executar o arquivo de *trace* do experimento anterior para validar qual bloco será substituído quando houver concorrência.

Validar o algoritmo **FIFO**:

Arquitetura: Architecture-06-R-s8-bs1-C1-tU-mA-s4-bs1-al2-wpWT-raFifo.xml

Trace: Trace06-R.txt

Tipo da execução: Passo a passo

Resultados que serão enfatizados:

* Enfatizar o **conjunto (*set*)** que é selecionado para alocar o novo bloco;
* Enfatizar a última instrução do arquivo de *trace* “2 4” que irá determinar a **substituição do primeiro bloco alocado no conjunto 0** devido à política de substituição **FIFO**.

Validar o algoritmo **LRU**:

Arquitetura: Architecture-07-R-s8-bs1-C1-tU-mA-s4-bs1-al2-wpWT-raLRU.xml

Trace: Trace06-R.txt

Tipo da execução: Passo a passo

Resultados que serão enfatizados:

* Enfatizar o **conjunto (*set*)** que é selecionado para alocar o novo bloco;
* Enfatizar a última instrução do arquivo de trace “2 4” que irá determinar a **substituição do bloco menos utilizado no conjunto 0** devido a política de substituição **LRU**.

1. Escrita de dados.

**Objetivo:** Compreender como funcionam as rotinas de escrita na memória *cache*.

**Detalhamento:** Utilizar um arquivo de arquitetura com uma memória *cache* pequena, com nível de **associatividade igual a dois** e com uma palavra por bloco. Executar um arquivo de *trace* que realiza carregamento de instruções, escrita e leitura de dados.

Arquitetura: Architecture-07-R-s8-bs1-C1-tU-mA-s4-bs1-al2-wpWT-raLRU.xml

Trace: Trace07-RW.txt

Tipo da execução: Passo a passo

Resultados que serão enfatizados:

* **Para um bloco ser escrito na memória *cache*, ele deve estar presente nela**. Caso não esteja presente, **o bloco deve ser carregado do nível inferior** através de uma leitura. Somente depois desta operação, a escrita poderá ser realizada na memória *cache*.

1. Política de atualização

**Motivação:** Após uma **escrita na memória *cache***, esta ficará com um **valor diferente da memória principal**. Como deve ser tratada esta inconsistência?

**Objetivo:** Compreender como funcionam as **políticas de escrita** *write-through* e *write-back* que têm a função de manter os dados da memória *cache* consistentes com os dados da memória principal.

**Detalhamento:** Utilizar um arquivo de arquitetura com uma memória *cache* pequena, com nível de **associatividade igual a dois** e com uma palavra por bloco. Executar um arquivo de *trace* que realiza carregamento de instruções, escrita e leitura de dados.

Validar a política ***Write-Through (WT)*:**

Arquitetura: Architecture-07-R-s8-bs1-C1-tU-mA-s4-bs1-al2-wpWT-raLRU.xml

Trace: Trace07-RW.txt

Tipo da execução: Passo a passo

Acessos de escrita na memória principal: 3

Validar a política ***Write-Back (WB)*:**

Arquitetura: Architecture-08-R-s8-bs1-C1-tU-mA-s4-bs1-al2-wpWB-raLRU.xml

Trace: Trace07-RW.txt

Tipo da execução: Passo a passo

Acessos de escrita na memória principal: 2

Resultados que serão enfatizados:

* **Para um bloco ser escrito na memória *cache*, ele deve estar presente nela**. Caso não esteja presente, **o bloco deve ser carregado do nível inferior** através de uma leitura. Somente depois desta operação, a escrita poderá ser realizada na memória cache.
* **Política WT**: **sempre atualiza a memória *cache* atual e o nível inferior**, mantendo os dados consistentes nos dois níveis;
* **Política WB**: atualiza a memória *cache* atual. No entanto, **só atualiza o nível inferior quando o bloco atualizado na memória *cache* atual necessita ser substituído**;
* Destacar o número de escritas que são realizadas na memória principal entre uma política e outra.

1. Caches Multiníveis

**Motivação:** Mesmo que ocorra uma falha na memória *cache*, demonstrar que é **possível agilizar o tempo total de execução inserindo uma *cache* rápida próxima ao processador e um outra cache um pouco maior, mas com uma taxa de acerto razoável**.

**Objetivo:** Demonstrar o funcionamento de ***caches* multiníveis** e o conceito da **penalidade por falta**.

**Detalhamento:**

1. Utilizarum arquivo de arquitetura com uma memória *cache* unificada, pequena e com nível de associatividade igual a um (mapeamento direto). Executar um arquivo de *trace* que realiza o carregamento de instruções estimulando a concorrência por blocos que serão alocados.
   1. Ressaltar o conceito da **penalidade por falta que é alta, pois em toda falha é necessário trazer um bloco do nível inferior**, neste caso a memória principal que tem um tempo de ciclo muito alto quando comparado com o tempo de ciclo da memória cache.

Arquitetura: Architecture-02-R-s8-bs1-C1-tU-mD-s4-bs1-al1-wpWT-raFifo.xml

Trace: Trace-08-1-R.txt

Tipo da execução: Direta

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memória / Taxas | Acessos | Total de acertos (***hit****s*) | Total de falhas (***mis****ses*) | **Tempo total** |
| *Cache* | 48 | 0 | **48** | **48** |
| Principal | 48 |  |  | **480** |
|  |  |  |  | **528** |

1. Utilizarum arquivo de arquitetura com uma memória *cache* em dois níveis, pequenas e com níveis de associatividade igual a um (mapeamento direto). Executar um arquivo de *trace* que realiza o carregamento de instruções estimulando a concorrência por blocos que serão alocados no primeiro nível da memória *cache*.
   1. Ressaltar o conceito da **penalidade por falta que é baixa, pois em toda falha é necessário trazer um bloco do nível inferior**, neste caso a memória cache de segundo nível que tem um tempo de ciclo muito baixo quando comparado com o tempo de ciclo da memória principal. **Mesmo com uma taxa de falhas alta no primeiro nível de cache, tem-se um bom desempenho do programa devido à baixa penalidade por falta**.

Arquitetura: Architecture-09-R-s8-bs1-C1-tU-mD-s2-bs1-al1-wpWB-raFifo-C2-tU-mD-s4-bs1-al1-wpWB-raFifo.xml

Trace: Trace-08-2-R.txt

Tipo da execução: Direta

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memória / Taxas | Acessos | Total de acertos (***hit****s*) | Total de falhas (***mis****ses*) | **Tempo total** |
| *Cache* L1 | 200 | 0 | **200** | **200** |
| *Cache* L2 | 200 | 196 | 4 | **200** |
| Principal | 4 |  |  | **40** |
|  |  |  |  | **440** |

Conceitos que serão enfatizados:

* Penalidade por falta;
* *Caches* multiníveis não possuem o objetivo de diminuir o número de falhas e sim diminuir a penalidade por falta;

Resultados que serão enfatizados:

Enfatizar que o desempenho do programa / *trace* aumentou, mesmo com um número de falhas bem maior no segundo experimento.

1. Caches Separadas

**Motivação:** Demonstrar que é possível eliminar a disputa entre a busca de dados e instruções em uma memória *cache*.

**Objetivo:** Demonstrar o funcionamento da *cache* separada em relação a *cache* unificada quanto a busca de instruções e dados.

**Detalhamento:**

1. Utilizarum arquivo de arquitetura com uma memória *cache* unificada, pequena e com nível de associatividade igual a um (mapeamento direto). Executar um arquivo de *trace* que realiza o carregamento de instruções, escrita e leitura de dados, estimulando a concorrência por blocos que serão alocados para instruções e dados.

Arquitetura: Architecture-02-R-s8-bs1-C1-tU-mD-s4-bs1-al1-wpWT-raFifo.xml

Trace: Trace09-RW.txt

Tipo da execução: Direta

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memória / Taxas | Acessos | Total de acertos (***hit****s*) | Total de falhas (***mis****ses*) | **Tempo total** |
| *Cache* | 6 | 1 | 5 | **7** |
| Principal | 6 |  |  | **70** |
|  |  |  |  | **77** |

1. Utilizarum arquivo de arquitetura com uma memória *cache* separada, pequena e com nível de associatividade igual a um (mapeamento direto). Executar um arquivo de *trace* que realiza o carregamento de instruções, escrita e leitura de dados, estimulando a concorrência por blocos que serão alocados para instruções e dados

Arquitetura: Architecture-10-R-s8-bs1-C1-tS-mD-s4-bs1-al1-wpWT-raFifo

Trace: Trace09-RW.txt

Tipo da execução: Direta

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memória / Taxas | Acessos | Total de acertos (***hit****s*) | Total de falhas (***mis****ses*) | **Tempo total** |
| *Data Cache* | 2 | 1 | 1 | **3** |
| *Instruction Cache* | 4 | 0 | 4 | **4** |
| Principal | 6 |  |  | **70** |
|  |  |  |  | **77** |

Resultados que serão enfatizados:

Enfatizar que não existe mais concorrência entre operações de carregamento, escrita / leitura de dados;

Políticas podem ser aplicadas especificamente à uma determinada *cache*, pois dados e instruções encontram-se separados.