**Laboratório VI – Memória Cache**

Nome: Gabriel Gatti da Silva Matrícula: 2021100336

Nome: Thiago F. N. Lahass Matrícula: 2021100178

Data: 02/02/23

**1. OBJETIVOS**

Entender o funcionamento do cache e de hierarquia de memória;

A influência da organização da cache no desempenho de um programa.

**2. ATIVIDADES**

**Atividade 1:**

*Altere os valores de array, size e max para que o programa, ao ser executado, crie um vetor de 100 posições, preenchido com valores aleatórios entre 0 e 9999. Quais são as chamadas de serviço responsáveis pela geração dos valores inteiros aleatórios? Qual foi a saída gerada?*

**SOLUÇÃO:**

Podemos alterar os valores de array, size e max da seguinte forma:

*array: .word 0 : 100 # an array of word, for storing values.*

*size: .word 100 # actual count of the elements in the array.*

*max: .word 9999 # Set upper bound to 10 (max\_int=2147483647)*

São feitas as seguintes chamadas de sistema no processo de geração de números aleatórios:

* Syscall 30: System Time syscall
* Syscall 40: Random seed
* Syscall 42: Random int range

A saída gerada foi:

*V= 1272 1697 3130 737 7917 7917 5524 3556 1162 8343 5949 3981 1587 8768 6374 4406 2013 9193 6799 1430 9035 6217 3823 1855 9460 6642 4248 2280 2280 9886 7067 4673 2705 312 7492 5099 9316 6923 4104 1711 9741 7348 168 7773 4955 2561 593 593 8198 5380 2986 7615 5222 2403 10 8041 5647 2829 435 8466 8466 6072 3254 860 8891 6498 3679 1286 2719 325 7506 5112 3144 751 7931 7931 5538 3569 1176 8356 5963 3994 1601 8781 6388 6388 1018 8624 5805 3412 1443 9049 6230 3837 1868 9474 6655 4262 2294 7081*

**Atividade 2:**

*Altere novamente os valores de array, size e max para que o programa, ao ser executado, crie um vetor de 33 posições, preenchido com valores aleatórios entre 0 e 99. Abra o componente “Tools → Memory Reference Visualizer” no MARS e clique em “Connect to MIPS”. Em seguida, abra o componente “Tools → Data Cache Simulator”. Observe os parâmetros para organização da cache na parte superior da janela deste último componente.*

*Configure uma pequena cache de 512 Bytes, com 32 blocos de 4 palavras, mapeamento direto e troca de palavras pela política LRU. Anote o número de acessos, o de acertos (hits) e falhas (misses) ocorrido. Compare esses resultados com a quantidade de lw e sw do código executado. Explique o que causou cada uma das falhas de cache na execução (não se desespere, sim, o número de falhas é muito pequeno!). Marque a opção “Runtime log” Enabled para facilitar sua resposta.*

**SOLUÇÃO:**

* Número de acessos: 889;
* Número acertos (hits): 871;
* Número de falhas (misses): 18.

Causa de cada uma das falhas de cache na execução:

1. *(1) address: 0x10010094 (tag 0x00080080) block range: 9-9 trying block 9 empty -- MISS*

Bloco de memória 9 vazio -> MISS;

1. *(13) address: 0x100100a0 (tag 0x00080080) block range: 10-10 trying block 10 empty -- MISS*

Bloco de memória 10 vazio -> MISS;

1. *(29) address: 0x100100b0 (tag 0x00080080) block range: 11-11 trying block 11 empty -- MISS*

Bloco de memória 11 vazio -> MISS;

1. *(39) address: 0x1001008c (tag 0x00080080) block range: 8-8 trying block 8 empty -- MISS*

Bloco de memória 8 vazio -> MISS;

1. *(46) address: 0x100100c0 (tag 0x00080080) block range: 12-12 trying block 12 empty -- MISS*

Bloco de memória 12 vazio -> MISS;

1. *(62) address: 0x100100d0 (tag 0x00080080) block range: 13-13 trying block 13 empty -- MISS*

Bloco de memória 13 vazio -> MISS;

1. *(78) address: 0x100100e0 (tag 0x00080080) block range: 14-14 trying block 14 empty -- MISS*

Bloco de memória 14 vazio -> MISS;

1. *(94) address: 0x100100f0 (tag 0x00080080) block range: 15-15 trying block 15 empty -- MISS*

Bloco de memória 15 vazio -> MISS;

1. *(110) address: 0x10010100 (tag 0x00080080) block range: 16-16 trying block 16 empty -- MISS*

Bloco de memória 16 vazio -> MISS;

1. *(127) address: 0x10010110 (tag 0x00080080) block range: 17-17 trying block 17 empty -- MISS*

Bloco de memória 17 vazio -> MISS;

1. *(137) address: 0x10010004 (tag 0x00080080) block range: 0-0 trying block 0 empty -- MISS*

Bloco de memória 0 vazio -> MISS;

1. *(183) address: 0x10010010 (tag 0x00080080) block range: 1-1 trying block 1 empty -- MISS*

Bloco de memória 1 vazio -> MISS;

1. *(263) address: 0x10010020 (tag 0x00080080) block range: 2-2 trying block 2 empty -- MISS*

Bloco de memória 2 vazio -> MISS;

1. *(343) address: 0x10010030 (tag 0x00080080) block range: 3-3 trying block 3 empty -- MISS*

Bloco de memória 3 vazio -> MISS;

1. *(423) address: 0x10010040 (tag 0x00080080) block range: 4-4 trying block 4 empty -- MISS*

Bloco de memória 4 vazio -> MISS;

1. *(503) address: 0x10010050 (tag 0x00080080) block range: 5-5 trying block 5 empty -- MISS*

Bloco de memória 5 vazio -> MISS;

1. *(583) address: 0x10010060 (tag 0x00080080) block range: 6-6 trying block 6 empty -- MISS*

Bloco de memória 6 vazio -> MISS;

1. *(663) address: 0x10010070 (tag 0x00080080) block range: 7-7 trying block 7 empty -- MISS*

Bloco de memória 7 vazio -> MISS;

**Atividade 3:**

Altere a organização da cache de acordo com os parâmetros indicados e preencha as linhas da tabela a seguir.

| **Número de**  **blocos** | **Tamanho do**  **bloco** | **Política de**  **troca** | **Política de**  **mapeamento** | **#**  **caminhos** | **#**  **acessos** | **#**  **hits** | **#**  **misses** | **Taxa média**  **de acerto** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **32** | **4** | **Random** | **Mapeamento**  **direto** | **1** | 889 | 871 | 18 | 98% |
| **32** | **4** | **LRU** | **Totalmente**  **associativo** | **32** | 889 | 871 | 18 | 98% |
| **32** | **4** | **LRU** | **Associativo**  **por conjunto** | **4** | 889 | 871 | 18 | 98% |
| **64** | **4** | **LRU** | **Mapeamento**  **direto** | **1** | 889 | 871 | 18 | 98% |
| **64** | **8** | **LRU** | **Mapeamento**  **direto** | **1** | 889 | 880 | 9 | 99% |
| **128** | **4** | **LRU** | **Mapeamento**  **direto** | **1** | 889 | 871 | 18 | 98% |
| **128** | **4** | **Random** | **Mapeamento**  **direto** | **1** | 889 | 871 | 18 | 98% |

**Atividade 4:**

*Usando o componente “Tools → Memory Reference Visualizer”, mostre qual é o padrão final dos acessos à memória de dados e à memória de código para a execução do código com os mesmos parâmetros da atividade anterior. Explique os padrões de cores gerados, usando como base (a) os acessos à memória de dados (instruções lw e sw) e à de instruções (PC+4) e (b) as cores associadas às células de memória (o parâmetro “Counter Value” associa o vermelho aos endereços com 10 ou mais acessos e o azul, a um único acesso durante a execução do programa).*

**SOLUÇÃO:**

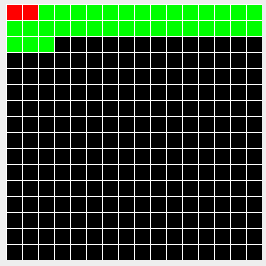
* Número de blocos: 32;

Tamanho do bloco: 4;

Política de troca: Random;

Política de mapeamento: Mapeamento direto;

Conjuntos: 1.

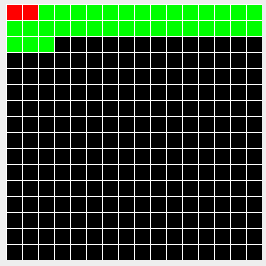
* Número de blocos: 32;

Tamanho do bloco: 4;

Política de troca: LRU;

Política de mapeamento: Totalmente associativo;

Conjuntos: 32.



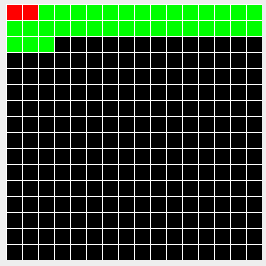
* Número de blocos: 32;

Tamanho do bloco: 4;

Política de troca: LRU;

Política de mapeamento: Associativo por conjunto;

Conjuntos: 4.



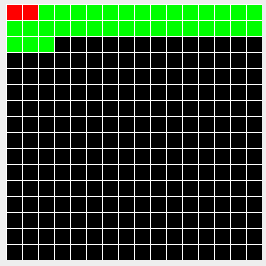
* Número de blocos: 64;

Tamanho do bloco: 4;

Política de troca: LRU;

Política de mapeamento: Mapeamento direto;

Conjuntos: 1.



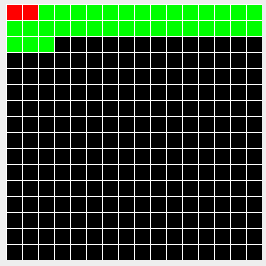
* Número de blocos: 64;

Tamanho do bloco: 8;

Política de troca: LRU;

Política de mapeamento: Mapeamento direto;

Conjuntos: 1.



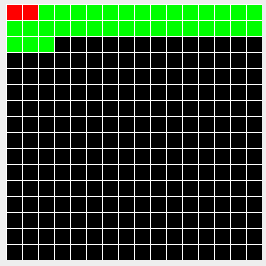
* Número de blocos: 128;

Tamanho do bloco: 4;

Política de troca: LRU;

Política de mapeamento: Mapeamento direto;

Conjuntos: 1.



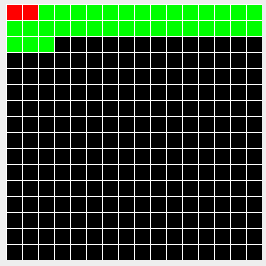
* Número de blocos: 128;

Tamanho do bloco: 4;

Política de troca: Random;

Política de mapeamento: Mapeamento direto;

Conjuntos: 1.



Portanto é possível observar que todas as configurações utilizaram basicamente a cache da mesma forma, isto é, a frequência com que cada posição da cache é acesa é similar. É visto que, as duas primeiras posições são as mais acessadas, por esse motivo estão em vermelho, e indicam que foram acessadas mais de 10 vezes durante a execução do programa. O "padrão" que pode ser observado é que, as outras posições utilizadas foram utilizadas cada uma duas vezes.

**Atividade 5:**

*Repita as 2 atividades anteriores para o programa Mult\_Matriz\_Inteiros 8x8.asm. Reduza a velocidade de execução e observe os acessos aos elementos das linhas e colunas de cada uma das matrizes usadas na multiplicação.*

**SOLUÇÃO:**

Altere a organização da cache de acordo com os parâmetros indicados e preencha as linhas da tabela a seguir (para o programa Mult Matriz Inteiros 8x8.asm).

| **Número de**  **blocos** | **Tamanho do**  **bloco** | **Política de**  **troca** | **Política de**  **mapeamento** | **#**  **caminhos** | **#**  **acessos** | **#**  **hits** | **#**  **misses** | **Taxa média**  **de acerto** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **32** | **4** | **Random** | **Mapeamento**  **direto** | **1** | 1361 | 1176 | 185 | 86% |
| **32** | **4** | **LRU** | **Totalmente**  **associativo** | **32** | 1361 | 2198 | 63 | 95% |
| **32** | **4** | **LRU** | **Associativo**  **por conjunto** | **4** | 1361 | 1301 | 60 | 96% |
| **64** | **4** | **LRU** | **Mapeamento**  **direto** | **1** | 1361 | 1312 | 49 | 96% |
| **64** | **8** | **LRU** | **Mapeamento**  **direto** | **1** | 1361 | 1336 | 25 | 98% |
| **128** | **4** | **LRU** | **Mapeamento**  **direto** | **1** | 1361 | 1312 | 49 | 96% |
| **128** | **4** | **Random** | **Mapeamento**  **direto** | **1** | 1361 | 1312 | 49 | 96% |

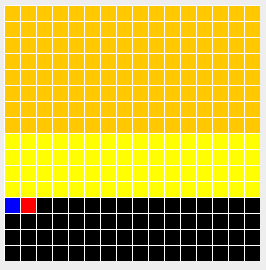
* Número de blocos: 32;

Tamanho do bloco: 4;

Política de troca: Random;

Política de mapeamento: Mapeamento direto;

Conjuntos: 1.



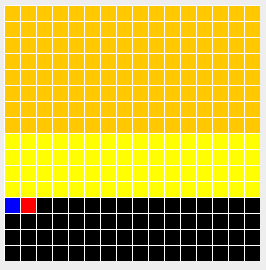
* Número de blocos: 32;

Tamanho do bloco: 4;

Política de troca: LRU;

Política de mapeamento: Totalmente associativo;

Conjuntos: 32.



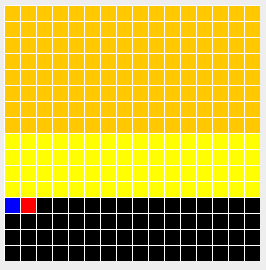
* Número de blocos: 32;

Tamanho do bloco: 4;

Política de troca: LRU;

Política de mapeamento: Associativo por conjunto;

Conjuntos: 4.



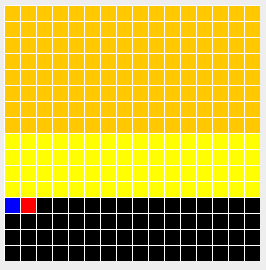
* Número de blocos: 64;

Tamanho do bloco: 4;

Política de troca: LRU;

Política de mapeamento: Mapeamento direto;

Conjuntos: 1.



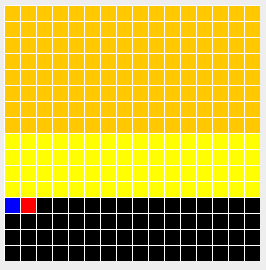
* Número de blocos: 64;

Tamanho do bloco: 8;

Política de troca: LRU;

Política de mapeamento: Mapeamento direto;

Conjuntos: 1.



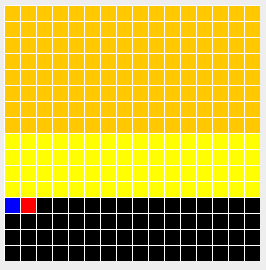
* Número de blocos: 128;

Tamanho do bloco: 4;

Política de troca: LRU;

Política de mapeamento: Mapeamento direto;

Conjuntos: 1.



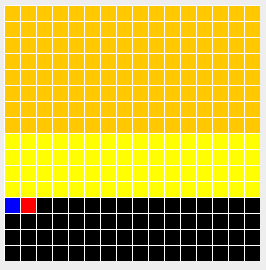
* Número de blocos: 128;

Tamanho do bloco: 4;

Política de troca: Random;

Política de mapeamento: Mapeamento direto;

Conjuntos: 1.



Portanto é possível observar que todas as configurações utilizaram basicamente a cache da mesma forma, isto é, a frequência com que cada posição da cache é acesa é similar. É visto que a posição em vermelho é a mais acessada, e indicam que foram acessadas mais de 10 vezes durante a execução do programa. Já a posição em azul é acessada somente uma vez.