Universidade Federal do Espírito Santo

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

Arquitetura de Computadores I – Turmas 01 e 02 (EARTE) – 2021/2
Prof. Rodolfo da Silva Villaça – rodolfo.villaca@ufes.br

Grupo: Dionatas Santos, Maria Julia Nolasco, Otávio Sales

Laboratório VIII – Memória Cache

1. Objetivos

- Entender o funcionamento do cache e de hierarquia de memória;
- A influência da organização da cache no desempenho de um programa.

2. Introdução

As memórias cache tiram vantagem das estatísticas de acesso para melhorar o desempenho de tempo de acesso. Programas não acessam RAM aleatoriamente. Os programas fazem acessos à posições de memória exibindo localidade temporal e espacial das palavras acessadas e as caches exploram esse comportamento para melhorar o desempenho da execução destes programas. A localidade temporal está baseada no princípio de que quando uma palavra é acessada, é provável que ela seja acessada novamente nos ciclos seguintes. A localidade espacial diz que, se um programa acessa uma palavra de memória, é provável que ele acesse as palavras vizinhas nos ciclos seguintes.

Para as etapas a seguir, você deverá utilizar o programa Vector_With_Random_Numbers.asm (anexo), que cria um vetor de N=10 posições no segmento de dados e, em seguida, preenche o vetor com valores aleatórios (no intervalo entre 0 e 10) ou com valores digitados pelo usuário. Em seguida, o programa imprime os valores de cada posição deste mesmo vetor.

3. Atividades

Após copiar e executar o programa com os valores dados, observe (e entenda) as funções dos valores array, size e max no programa.

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?

```
addi $v0, $zero, 40        # Syscall 40: Random seed

addi $v0, $zero, 42        # Syscall 42: Random int range

syscall                    # Generate a random number and put it in $a0
```

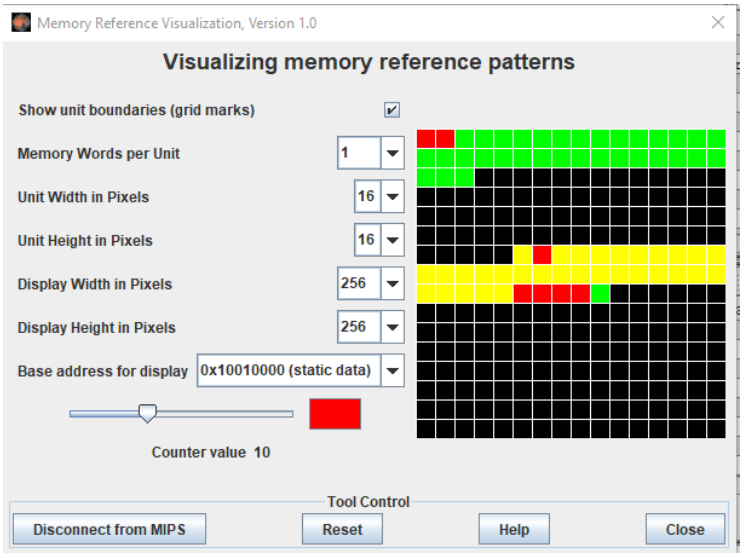
```
Input value#1: 6152
Input value#2: 6152
Input value#3: 6152
Input value#4: 6152
Input value#5: 6152
Input value#6: 6152
Input value#7: 6152
Input value#8: 6152
Input value#9: 6152
Input value#10: 6152
Input value#11: 6152
Input value#12: 6152
```

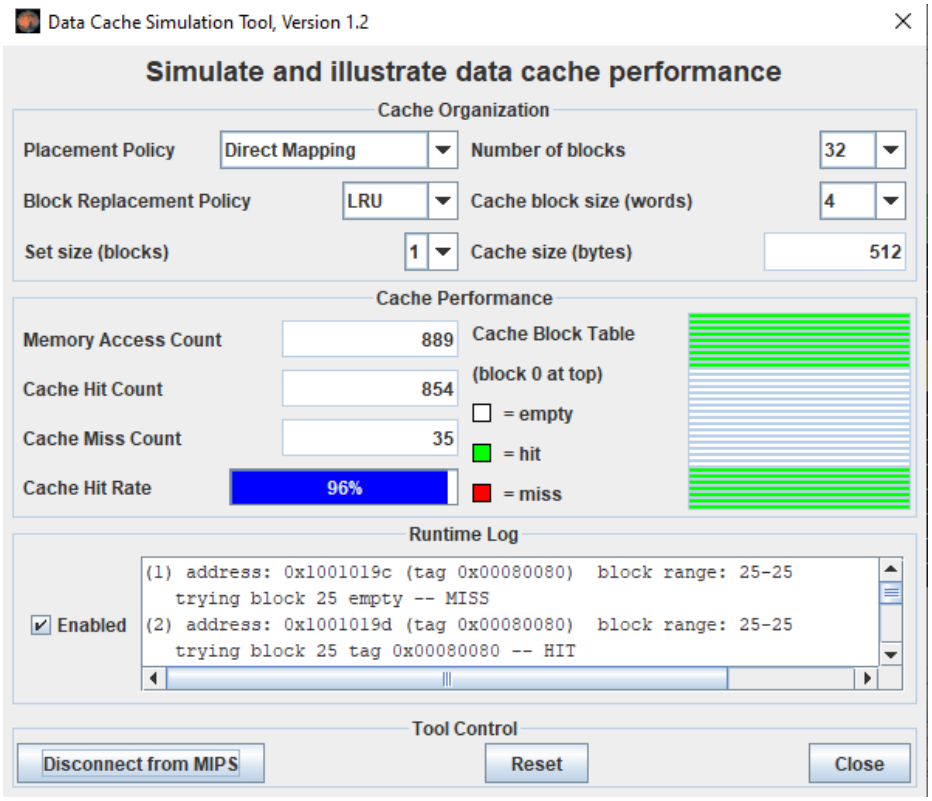
Input value#13: 6152
Input value#14: 6152
Input value#15: 6152
Input value#16: 6152
Input value#17: 6152
Input value#18: 6152
Input value#19: 6152
Input value#20: 6152
Input value#21: 6152
Input value#22: 6152
Input value#23: 6152
Input value#24: 6152
Input value#25: 6152
Input value#26: 6152
Input value#27: 6152
Input value#28: 6152
Input value#29: 6152
Input value#30: 6152
Input value#31: 6152
Input value#32: 6152
Input value#33: 6152
Input value#34: 6152
Input value#35: 6152
Input value#36: 6152
Input value#37: 6152
Input value#38: 6152
Input value#39: 6152
Input value#40: 6152
Input value#41: 6152
Input value#42: 6152
Input value#43: 6152
Input value#44: 6152
Input value#45: 6152
Input value#46: 6152
Input value#47: 6152
Input value#48: 6152
Input value#49: 6152
Input value#50: 6152
Input value#51: 6152
Input value#52: 6152
Input value#53: 6152
Input value#54: 6152
Input value#55: 6152
Input value#56: 6152
Input value#57: 6152
Input value#58: 6152
Input value#59: 6152
Input value#60: 6152
Input value#61: 6152
Input value#62: 6152
Input value#63: 6152
Input value#64: 6152
Input value#65: 6152
Input value#66: 6152
Input value#67: 6152
Input value#68: 6152
Input value#69: 6152
Input value#70: 6152

Input value#71: 6152
Input value#72: 5296
Input value#73: 5296
Input value#74: 5296
Input value#75: 5296
Input value#76: 5296
Input value#77: 5296
Input value#78: 5296
Input value#79: 5296
Input value#80: 5296
Input value#81: 5296
Input value#82: 5296
Input value#83: 5296
Input value#84: 5296
Input value#85: 5296
Input value#86: 5296
Input value#87: 5296
Input value#88: 5296
Input value#89: 5296
Input value#90: 5296
Input value#91: 5296
Input value#92: 5296
Input value#93: 5296
Input value#94: 5296
Input value#95: 5296
Input value#96: 5296
Input value#97: 5296
Input value#98: 5296
Input value#99: 5296
Input value#100: 5296

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 o número de lw e sw do código executado. Explique o que causou cada uma das falhas de cache na execução (não se desespere porque o número de falhas é muito pequeno!). Marque a opção "*Runtime log*" *Enabled* para facilitar sua resposta.





HITS = 854

MISS = 35

sw = 1

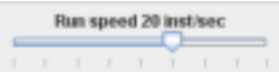
lw = 4

Atividade 3: Altere a organização da cache de acordo com os parâmetros indicados e preencha as linhas da tabela a seguir.

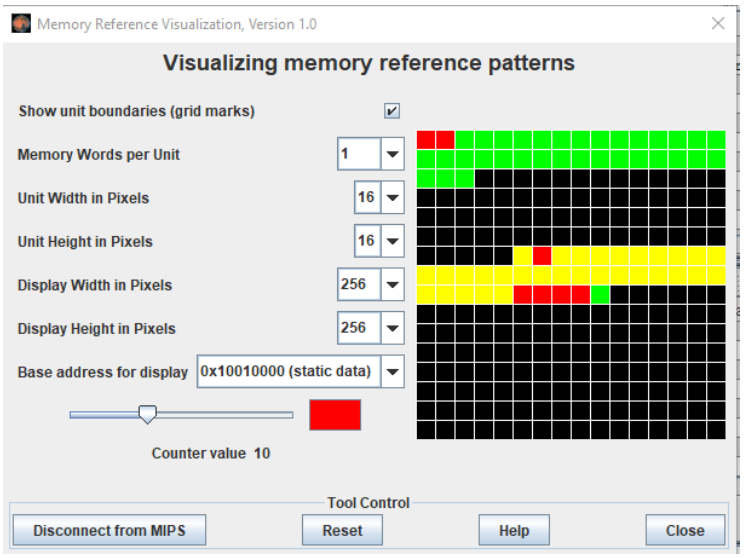
Número de blocos	Tamanho do bloco	Política de troca	Política de mapeamento	# caminhos	# acessos	# hits	# misses	Taxa média de acerto
32	4	Random	Mapeamento direto	1	1778	1721	57	97%
32	4	LRU	Totalmente associativo	32	889	870	19	98%

32	4	LRU	Associativo por conjunto	4	889	870	19	98%
64	4	LRU	Mapeamento direto	1	889	870	19	98%
64	8	LRU	Mapeamento direto	1	889	878	11	99%
128	4	LRU	Mapeamento direto	1	889	870	19	98%
128	4	Random	Mapeamento direto	1	889	870	19	98%

Para facilitar o entendimento do que acontece durante a execução do programa, reduzindo a velocidade de execução (instruções por segundo) clicando no componente abaixo da interface do MARS:



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 Q2. 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).



vermelho = MISS (?)

verde = HIT (?)

amarelo = ?

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.

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	1298	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%