Centro Federal de Educação Tecnológica de Minas Gerais

Laboratório de Arquitetura e Organização de Computadores II

Igor Luciano de Paula e Eduardo Junqueira de Matos

Prática I

6 de abril de 2018

1 Memória RAM

- Memória RAM implementada utilizando a biblioteca LPM. A leitura e escrita foram realizadas utilizando o display de 7-segmentos.

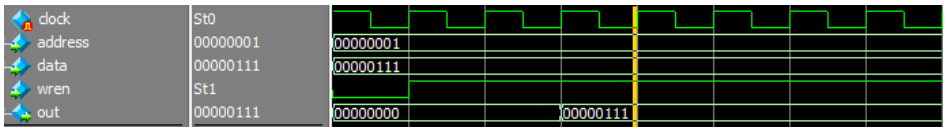

Figura 1: Escrita na memória RAM.

Conforme apresentado na figura 1, é possível visualizar o processo de escrita na memória RAM. outuando o sinal presente na variável ***wren*** recebe nível lógico alto, é habilitada a escrita na memória, outue ocorre sempre na borda de subida do Clock (variável ***clock***).
Como demonstrado na simulação[1], quando ***wren*** em nível lógico alto e o ***clock*** na borda de subida, o dado presente na variável ***data*** é escrito na memória no endereço presente em ***adress***, e demonstrado na variável de saída ***out***.

[1] Todas as simulações foram realizadas utilizando o software ModelSIM.

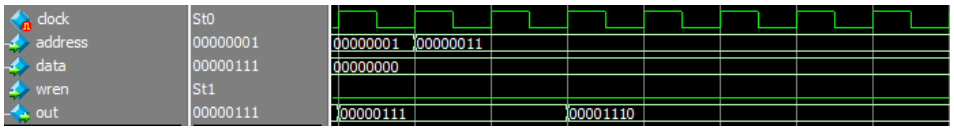

Figura 2: Leitura na memória RAM.

É possível verificar na figura 2, o processo de leitura na memória RAM. Quando o sinal presente em ***wren*** se mantém em nível lógico baixo ocorre a leitura da memória, utilizando a borda de descida do Clock (***clock***).

Como demonstrado na segunda simulação (figura 2), na ocasião em que ***wren*** esteja em nível lógico baixo e o ***clock*** na borda de descida, o dado (***data***) presente no endereço (***adress***) é exibido na variável de saída ***out***, realizando a leitura do endereço desejado.

2 Memory Initialization File - MIF

- A memória RAM implementada utilizando inicialização de memória por um arquivo MIF (***memory initialization file*** - arquivo de inicialização de memória). A leitura e escrita foram realizadas utilizando o display de 7-segmentos

```
WIDTH=8;
DEPTH=256;

ADDRESS_RADIX=UNS;
DATA_RADIX=UNS;

CONTENT
BEGIN
        [0..4]    :   242;
        5         :   13;
        6         :   170;
        [7..255]  :   0;
END;
```

Figura 3: Arquivo MIF.

A figura 3 apresenta o aroutuivo MIF utilizado para inicializar a memória. As primeiras 5 posições **[0-4]** da memória foram inicializadas com o valor em Decimal '**242**'(binário: 11110010). O endereço seguinte da memória **[5]** foi inicializado com o dado '**13**'(binário: 00001101). O endereço de memória **[6]** foi inicializado com o dado '**170**'(binário: 10101010). O restante das posições da memória foram inicializadas com '**0**'.

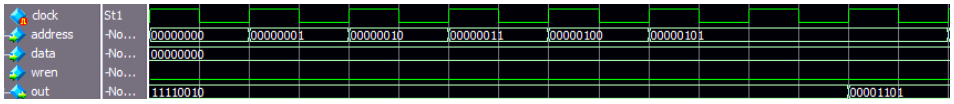

Figura 4: Estado inicial da memória.

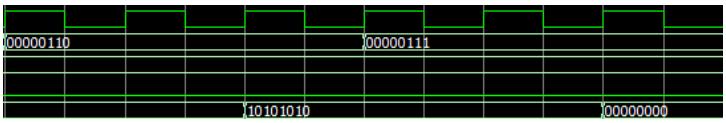

Figura 5: Continuação da Figura 4.

As figuras 4 e 5 apresentam todos os estados iniciais da memória. Da posição **00000000** até a posição **00000111** (**[0-7]**) o valor lido e apresentado na saída ***out*** foi **11110010** (**242** em Decimal). Para a posição **00000101** (**[5]**) o valor de saída foi **00001101** (**13** em Decimal). A posição **00000110** (**[6]**) teve o valor de saída **10101010** (**170** em Decimal)Já para **00000111**(**[7]**), o endereço foi estanciado com '**0**' (assim como os valores subsequentes), apresentando a saída **00000000**.

3 Hierarquia de Memória

- Implementação de uma hierarquia de memória organizada em uma Cache L1 e uma memória principal (atualização da memória utilizando Write-Back). A Cache L1 é totalmenta associativa e a memória principal é diretamente mapeada.

Foi implementada uma Cache de quatro blocos, com funcionalidade de write-back (um dado escrito é atualizado na RAM quando sua posição é sobrescrita na Cache) e com decisão LRU (***Least Recently Used*** - Menos recentemente usado) o bloco menos recentemente acessado está sujeita a ser sobrescrito. Para tanto, foi criado um arranjo bidimensional de registradores, com 16 bits cada. A figura 6 ilustra a distribuição dos bits da Cache.

BLOCOS	DADO				TAG				
	0	1	2	3	4	5	6	7	8
0									
1									
2									
3									

			LRU		DIRTY	VALIDADE
9	10	11	12	13	14	15

Figura 6: Distribuição de bits.

O esquema de interconexão entre os módulos, para a implementação da hierarquia, foi realizado conforme a figura 7.

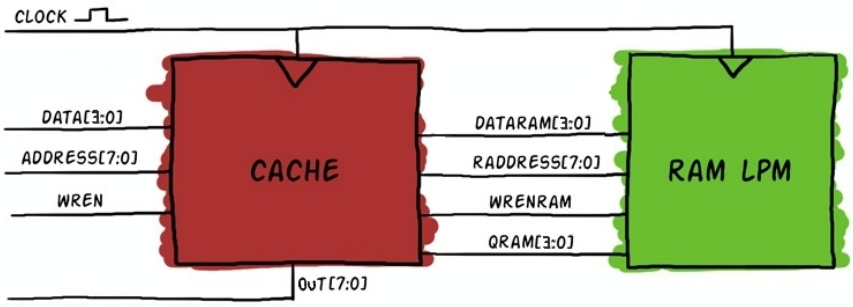

Figura 7: Esquema de ligação.

O funcionamento da Cache consiste em uma máquina de estados, conforme a figura 8. A cada borda de subida do clock, ela se encontra em um e apenas um estado, e é processado. Um novo estado é então assumido para ser executado no próximo ciclo. Enquanto uma condição de hit demanda um ciclo para retornar ou escrever o dado pedido, um caso de miss precisa de mais ciclos para acessar a RAM e tomar a mesma ação.

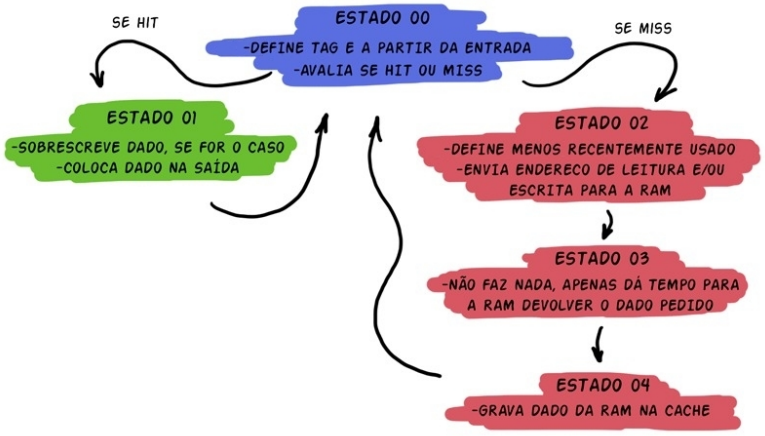

Figura 8: Estados.

- Simulações:

```
Entradas:
        - SW[17:10]: Adress
        - SW[7:0]: Data
        - SW[8]: Wren
        - SW[9]: Clock

Saídas:
        - HEX[0]: Unidade
        - HEX[1]: Dezena
        - HEX[2]: Centena
```

Figura 9: Entadas e saídas utilizadas - Cache.

Da figura 10 até a figura 13 monstra os estados iniciais da memória Cache.

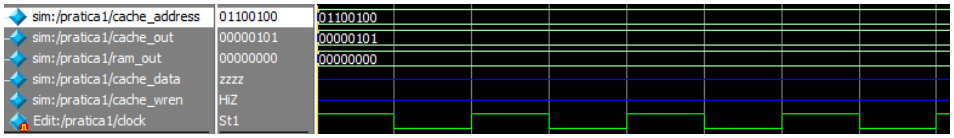

Figura 10: Leitura da posição 0.

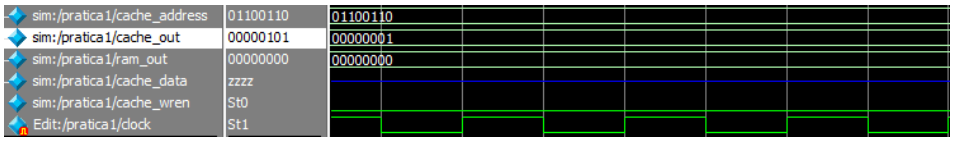

Figura 11: Leitura da posição 1.

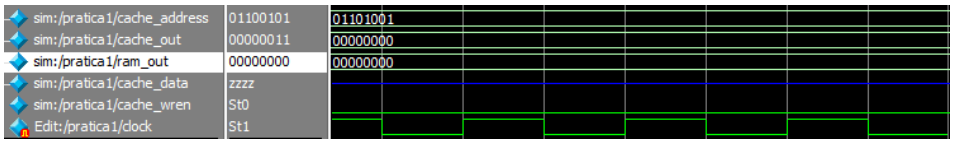

Figura 12: Leitura da posição 2.

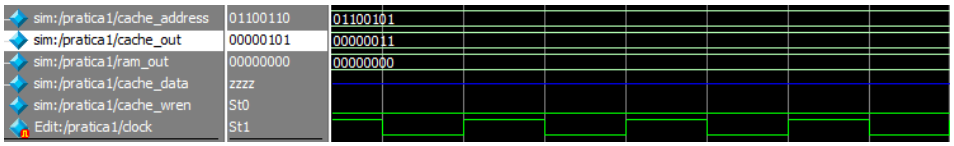

Figura 13: Leitura da posição 3.

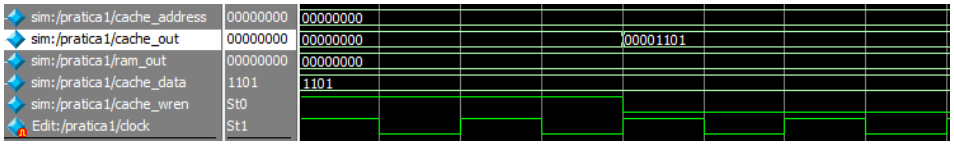

Figura 14: Escrita e leitura da memória Cache (i)

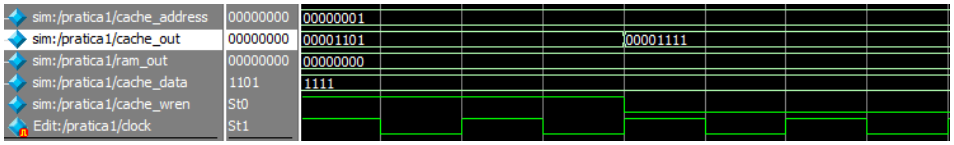

Figura 15: Escrita e leitura da memória Cache (ii)

Como apresentado na demonstração em sala de aula, a memória Cache estava apresentando erros na busca de dados na memória RAM(quando o dado é requerido na memória Cache e não encontrado - miss - não estava sendo possível buscá-lo na memória RAM). Entretanto, as demais funções implementadas estavam funcionando perfeitamente, como write-back, leitura, escrita, utilização e atualização do LRU e etc.