

[Aula 14] Hierarquia de memória 1: *Introdução*

Prof. João F. Mari
joaof.mari@ufv.br

Aula 14 - Hierarquia de memória 1

Bibliografia

1. PATTERSON, D.A; HENNESSY, J.L. **Organização e Projeto de Computadores: A Interface Hardware/Software**. 3a. Ed. Elsevier, 2005.
 - Capítulo 7.
2. Notas de aula do prof. Luciano J. Senger:
 - <http://www.ljsenger.net/classroom.html>
3. Notas de aula da profa. Mary Jane Irwin
 - CSE 331 Computer Organization and Design
 - <http://www.cse.psu.edu/research/mdl/mji/mjicourses>



Hierarquia de memória

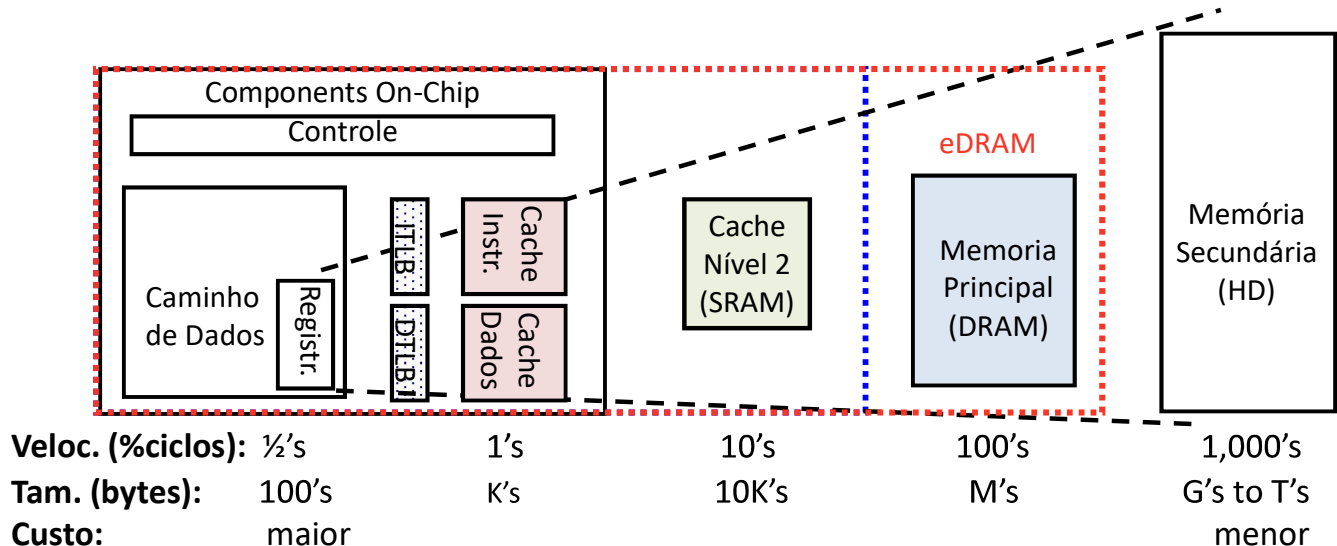
- Memórias rápidas são pequenas (porque são caras) e memórias grandes são lentas.
- Como criar a ilusão de uma memória grande, rápida e barata?
 - Hierarquia de memórias.

Princípio da localidade

- Localidade temporal
 - Se um item [da memória] é referenciado, ele tenderá a ser referenciado novamente em breve;
 - **[EX]** *A maioria dos programas contém loops:*
 - *Instruções e dados são acessados de modo repetitivo.*
- Localidade espacial
 - Se um item [da memória] é referenciado, os itens cujos endereços estão próximos tenderão a ser referenciados em breve;
 - **[Ex]** *Instruções são acessadas sequencialmente.*

Princípio da localidade

- Princípio da localidade
 - Proporcionar ao usuário a **maior** quantidade de memória possível com a tecnologia **mais barata...**
 - Com a velocidade oferecida pela tecnologia mais rápida (e cara).



Hierarquia de memória

- Múltiplos níveis de memória com diferentes velocidades e tamanhos
 - Memórias mais rápidas são mais caras por bit
- Tecnologias principais
 - DRAM (*Dinamic Random Access Memory*)
 - Memória principal
 - SRAM (*Static Random Access Memory*)
 - Níveis mais próximos do processador (caches)
 - Disco Magnético
 - Maior e mais lento nível da hierarquia.
 - Atualmente dividem espaço com os SSDs (*Solid State Disks*).

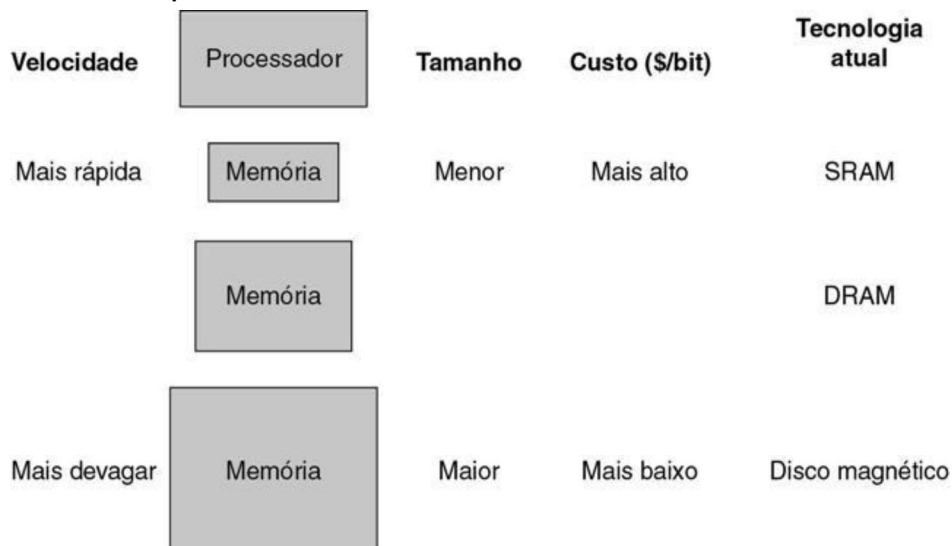
Hierarquia de memória

- O tempo de acesso e preço por bit variam muito entre as tecnologias de memória.
 - Valores típicos de 2008, quando foi escrita a última versão do livro texto (4 edição).

Tecnologia de memória	Tempo de acesso típico	US\$ por GB em 2008
SRAM	0,5 a 2,5 ns	2000 a 5000
DRAM	50 a 70 ns	20 a 75
Disco Magnético	5.000.000 a 20.000.000 ns	0,20 a 2

Hierarquia de memória

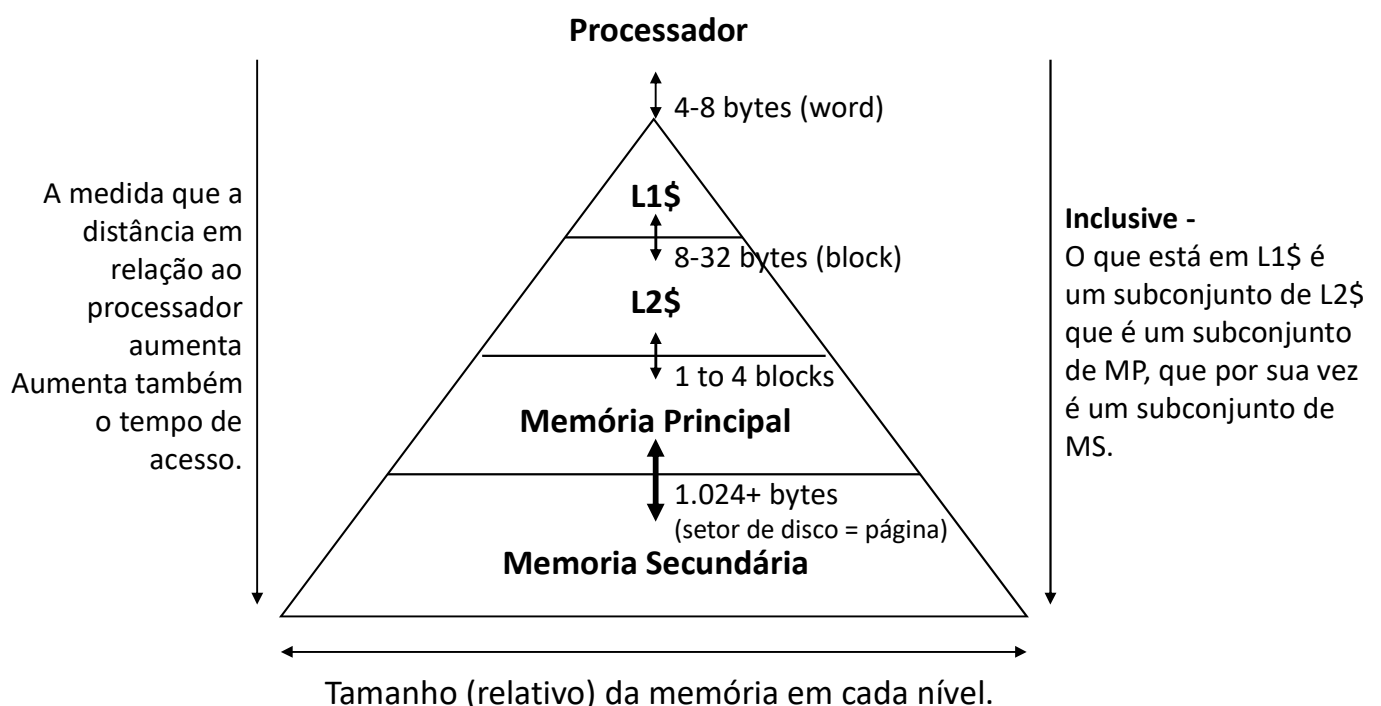
- Estrutura básica de uma hierarquia de memória.**
 - Implementar o sistema de memória como uma hierarquia faz com que o usuário tenha a ilusão de uma memória que é tão grande quanto o maior nível da hierarquia, mas que pode ser acessada como se fosse construída com a memória mais rápida.



Organização da hierarquia de memória

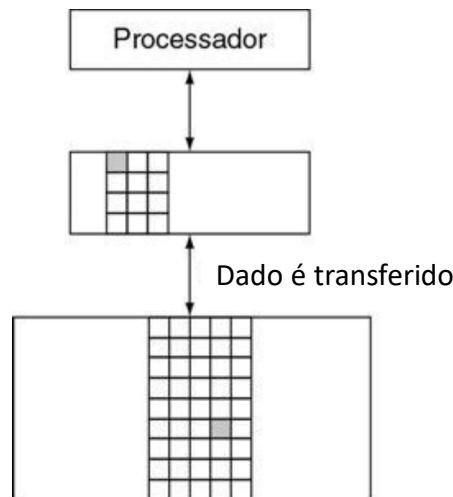
- Um nível mais próximo do processador em geral é um subconjunto de qualquer nível mais distante.
- Todos os dados estão armazenados no nível mais distante (mais baixo).
- Conforme nos afastamos do processador, os níveis levam cada vez mais tempo para serem acessados.
- Dados são copiados apenas entre dois níveis ao mesmo tempo:
 - A informação mínima que pode estar presente ou ausente em um hierarquia de dois níveis.

Estrutura da hierarquia de memória



Organização da hierarquia de memória

- Cada par de níveis na hierarquia de memória pode ser imaginado como tendo um nível superior e um nível inferior.
 - Dentro de cada nível, a unidade de informação que está presente ou não é chamada de um bloco ou uma linha.
 - Em geral, transferimos um bloco inteiro quando copiamos algo entre os níveis.



Organização da hierarquia de memória

- **ACERTO:**
 - Os dados requisitados pelo processador estão em algum bloco no nível superior.
- **FALHA:**
 - Os dados não forem encontrados no nível superior;
 - O nível inferior precisa ser acessado para recuperar o bloco com os dados requisitados;
 - **Taxa de acertos:** fração dos acessos a memória encontrados no nível superior
 - **Taxa de falhas:** $1 - \text{taxa de acertos}$
 - **Tempo de acerto:** Tempo necessário para acessar o nível superior da hierarquia
 - **Penalidade de falta:** tempo para substituir um bloco no nível superior pelo bloco correto do nível superior

Gerenciamento da hierarquia de memória

- Registradores \leftrightarrow Memória:
 - Compilador / Programador.
- Cache \leftrightarrow Memória Principal:
 - Controlador de cache (hardware).
- Memória Principal \leftrightarrow Disco Rígido:
 - Sistema Operacional (Memória Virtual);
 - Mapeamento (endereços virtuais \rightarrow endereços físicos) é assistido pelo hardware (TLB);
 - Programador (arquivos).

Cache: um lugar seguro para esconder ou guardar coisas.
Webster's New World Dictionary of the American Language

PRINCIPIOS BÁSICOS DE CACHE

Princípios básicos de cache

- **Cache:**
 - Nível(s) de hierarquia entre o processador (registradores) e a memória principal.
- Como saber se os dados estão no cache e como encontrá-los?
 - Mapeamento direto:
 - Cada local da memória é mapeada diretamente para um local exato do cache:
 - $(\text{endereço do bloco}) \% (\text{número de blocos na cache})$
 - Se o número de entradas na cache é potência de 2:
 - Posição na cache = $\log_2(\text{tamanho da cache em blocos})$ bits menos significativos
 - **[EX]** *As posições em uma cache com oito blocos são endereçadas com os três (2^3) bits menos significativos da endereço.*

Princípios básicos de cache

- **Uma cache diretamente mapeada com oito entradas mostrando os endereços das palavras de memória entre 0 e 31 que são mapeadas para os mesmos locais de cache.**
 - Como há oito palavras na cache, um endereço X é mapeado para a palavra de cache $X \bmod 8$.
 - Ou seja, os $\log_2(8) = 3$ bits menos significativos são usados como o índice da cache.
 - Assim, os endereços 00001_2 , 01001_2 , 10001_2 e 11001_2 são todos mapeados para a entrada 001_2 da cache, enquanto os endereços 00101_2 , 01101_2 , 10101_2 e 11101_2 são todos mapeados para a entrada 101_2 da cache.

Princípios básicos de cache

- Como saber se o dado na cache é o dado solicitado?
 - Incluir tags na cache
 - Tag: os bits do endereço que não foram utilizados como índice para a cache.
- Pode ser que a cache não possua dados válidos
 - No início da execução algumas posições da cache podem conter dados inválidos.
 - 1 bit de validade
 - 1: OK
 - 0: posição com dados inválidos

Acessando uma cache

O conteúdo da cache é mostrado para cada requisição de referência que falha, com os campos índice e tag mostrados em binário.

- A cache inicialmente está vazia, com todos os bits de validade (entrada V da cache) inativos (N). (a)
- O processador requisita os seguintes endereços:
 - 10110_2 (falha), (b)
 - 11010_2 (falha), (c)
 - 10110_2 (acerto),
 - 11010_2 (acerto),
 - 10000_2 (falha), (d)
 - 00011_2 (falha), (e)
 - 10000_2 (acerto) e
 - 10010_2 (falha). (f)
- As figuras mostram o conteúdo da cache após cada falha na sequência ter sido tratada.
- Quando o endereço 10010_2 (18) é referenciado, a entrada para o endereço 11010_2 (26) precisa ser substituída, e uma referência a 11010_2 causará uma falha subsequente.
- O campo tag conterá apenas a parte superior do endereço.

Índice	V	Tag	Data	Index	V	Tag	Data
000	N			000	N		
001	N			001	N		
010	N			010	N		
011	N			011	N		
100	N			100	N		
101	N			101	N		
110	N			110	Y	10_{two}	Memory (10110_{two})
111	N			111	N		

a. O estado inicial do cache depois de power-on

b. Depois de lidar com uma falha de endereço (10110_{two})

Índice	V	Tag	Data	Índice	V	Tag	Data
000	N			000	Y	10_{two}	Memory (10000_{two})
001	N			001	N		
010	Y	11_{two}	Memory (11010_{two})	010	Y	11_{two}	Memory (11010_{two})
011	N			011	N		
100	N			100	N		
101	N			101	N		
110	Y	10_{two}	Memory (10110_{two})	110	Y	10_{two}	Memory (10110_{two})
111	N			111	N		

c. Depois de lidar com uma falha de endereço (11010_{two})d. Depois de lidar com uma falha de endereço (10000_{two})

Índice	V	Tag	Data	Índice	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})	000	Y	10_{two}	Memory (10000_{two})
001	N			001	N		
010	Y	11_{two}	Memory (11010_{two})	010	Y	10_{two}	Memory (10010_{two})
011	Y	00_{two}	Memory (00011_{two})	011	Y	00_{two}	Memory (00011_{two})
100	N			100	N		
101	N			101	N		
110	Y	10_{two}	Memory (10110_{two})	110	Y	10_{two}	Memory (10110_{two})
111	N			111	N		

e. Depois de lidar com uma falha de endereço (00011_{two})f. Depois de lidar com uma falha de endereço (10010_{two})

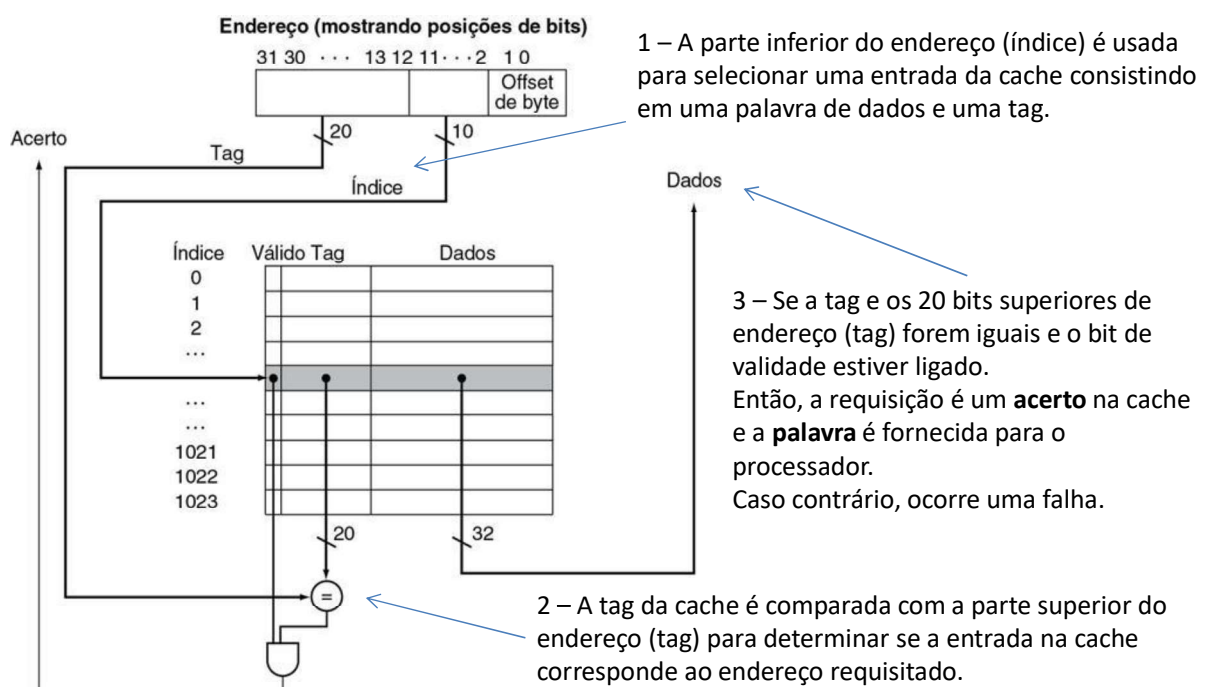
Acessando uma cache

- Ação para cada referência:

Endereço decimal da referência	Endereço binário da referência	Acerto ou falha na cache	Bloco de cache atribuído (onde foi encontrado ou inserido)
22	10110 ₂	falha	$(10110_2 \bmod 8) = 110_2$
26	11010 ₂	falha	$(11010_2 \bmod 8) = 010_2$
22	10110 ₂	acerto	$(10110_2 \bmod 8) = 110_2$
26	11010 ₂	acerto	$(11010_2 \bmod 8) = 010_2$
16	10000 ₂	falha	$(10000_2 \bmod 8) = 000_2$
3	00011 ₂	falha	$(00011_2 \bmod 8) = 011_2$
16	10000 ₂	acerto	$(10000_2 \bmod 8) = 000_2$
18	10010 ₂	falha	$(10010_2 \bmod 8) = 010_2$

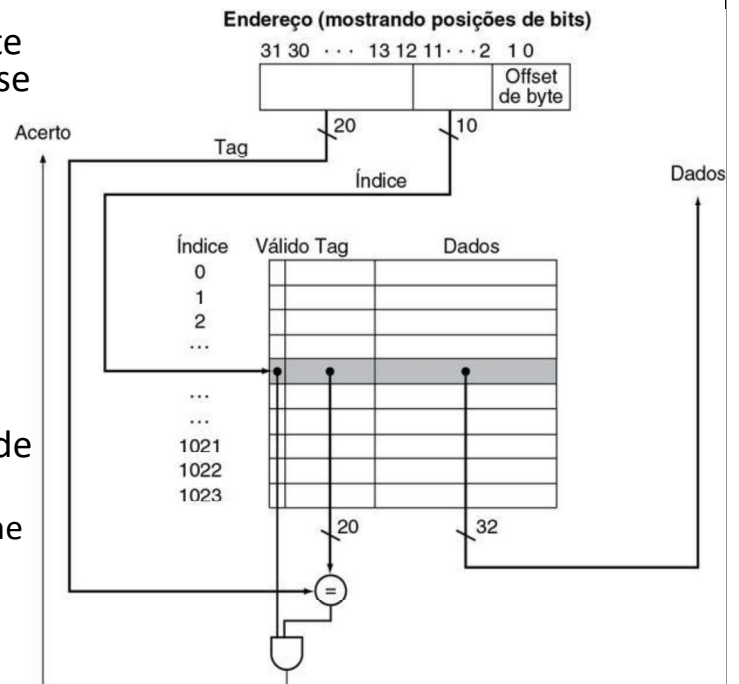
Acessando uma cache

A cache possui 2^{10} (1024) palavras e o tamanho de bloco é de 1 palavra. Dessa forma, 10 bits são usados para indexar a cache, deixando 20 bits ($32 - 10 - 2$) para serem comparados com a tag.



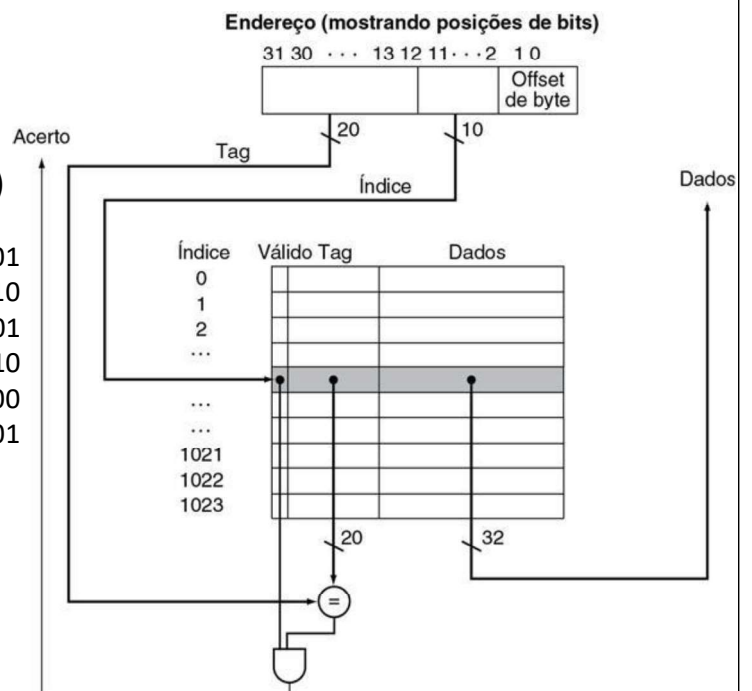
Acessando uma cache

- Para esta cache, a parte inferior do endereço é usada para selecionar uma entrada da cache consistindo em uma palavra de dados e uma tag.
- A tag da cache é comparada com a parte superior do endereço para determinar se a entrada na cache corresponde ao endereço requisitado.
- A cache tem 2^{10} (1024) palavras e um tamanho de bloco de 1 palavra:
 - 10 bits são usados para indexar a cache
 - Deixando $32 - 10 - 2 = 20$ bits para serem comparados com a tag.
- Se a tag e os 20 bits superiores de endereço forem iguais e o bit de validade estiver ligado
 - Então, a requisição é um acerto na cache e a palavra é fornecida para o processador.
 - Caso contrário, ocorre uma falha.



Exercício

- Considere a cache do exemplo anterior:
 - Palavras de 32 bits;
 - 1024 palavras;
 - 1 palavra por bloco;
 - 1024 blocos.
- 1) Explique por que o índice possui 10 bits.
- 2) Explique por que a Tag possui 20 bits.
- 3) Simule (conforme o exemplo da pag. 19) as seguintes solicitações:
 - #1) 0101 1010 0110 0111 1000 1111 1010 0001
 - #2) 0001 0101 0011 0101 0110 0001 0000 1010
 - #3) 0101 0101 0110 0111 1000 1111 1010 0001
 - #4) 0001 0101 0011 0101 0110 0001 0000 1010
 - #5) 0101 1010 1110 0001 0001 1110 0101 1100
 - #6) 0101 0101 0110 0111 1000 1111 1010 0001
- Desenhe a cache após cada solicitação.
- Informe se ocorreu um acerto ou falha.



Bibliografia

1. PATTERSON, D.A; HENNESSY, J.L. **Organização e Projeto de Computadores: A Interface Hardware/Software**. 3a. Ed. Elsevier, 2005.
 - Capítulo 7.
2. Notas de aula do prof. Luciano J. Senger:
 - <http://www.ljsenger.net/classroom.html>
3. Notas de aula da profa. Mary Jane Irwin
 - CSE 331 Computer Organization and Design
 - <http://www.cse.psu.edu/research/mdl/mji/mjicourses>



FIM

- FIM:
 - **Aula 14** – Hierarquia de memória 1.
- Próxima aula:
 - **Aula 15** – Hierarquia de memória 2 – Memória cache.