Arquiteturas Paralelas e Distribuídas

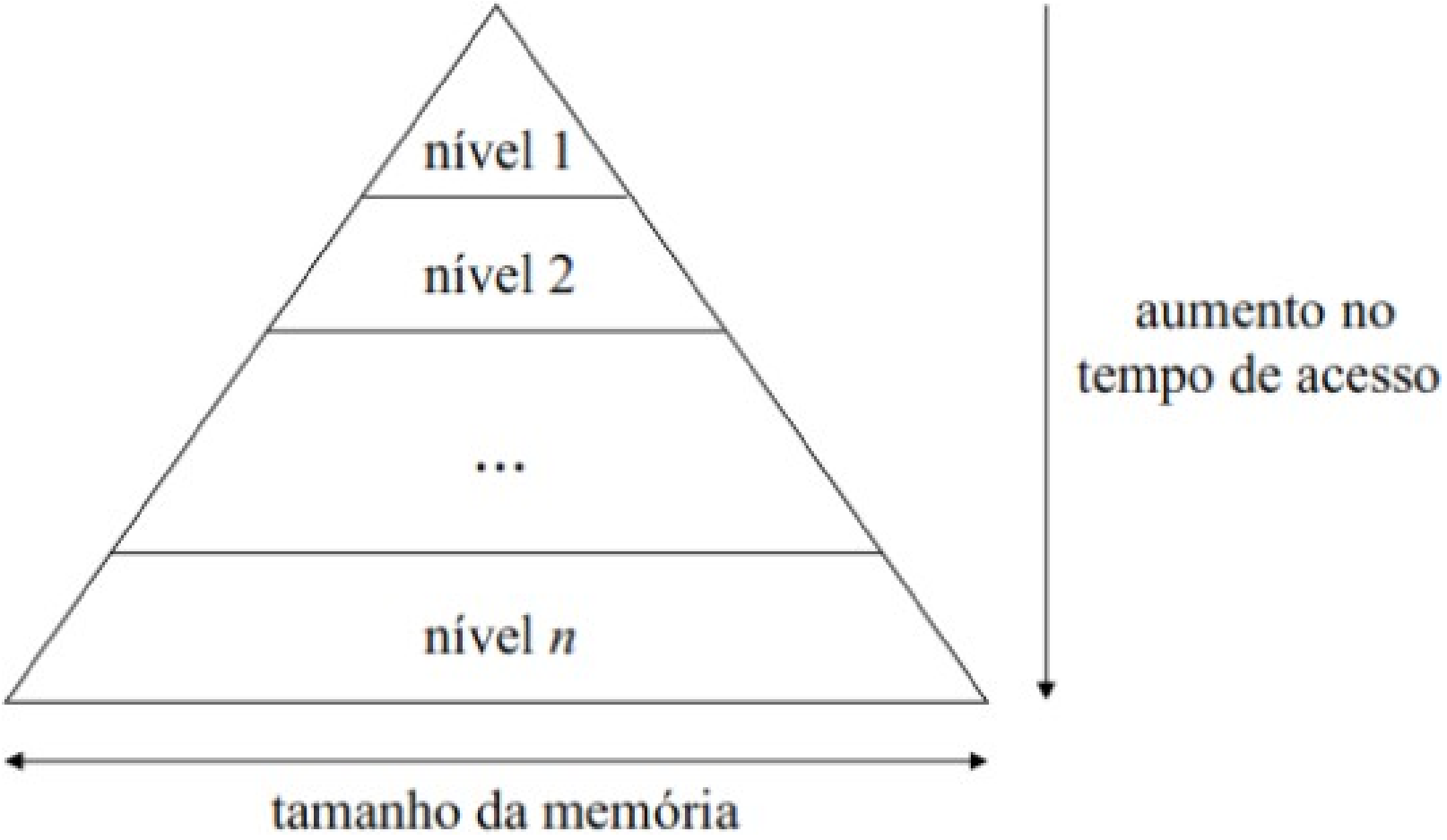
Memória distribuída e hierarquia de memórias

Eduardo Furlan Miranda

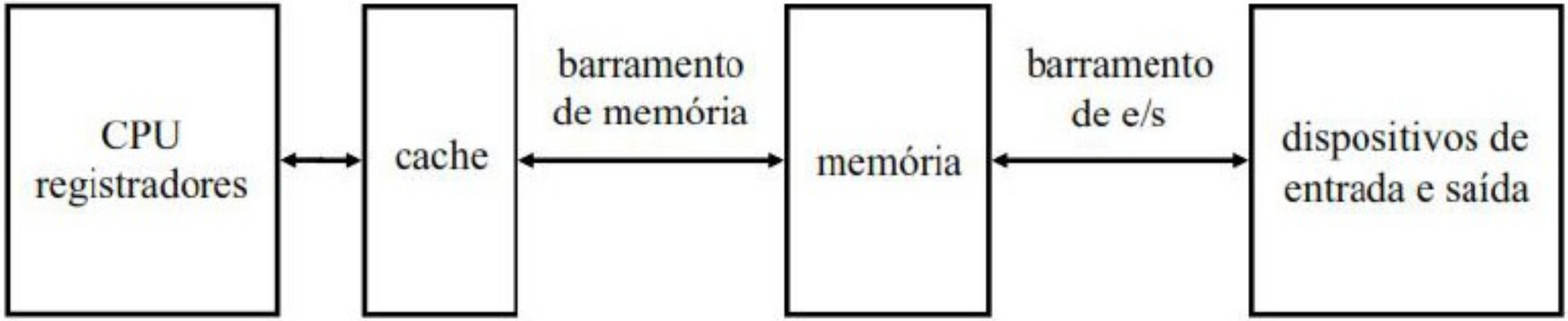
Baseado em: LIMA, E. Fundamentos da Programação Paralela. 2018.

Hierarquia de memória

- A hierarquia de memória organiza a memória em diferentes níveis, com velocidades e capacidades variadas



- Os níveis da hierarquia são subconjuntos, ou seja, dados de um nível também estão presentes nos níveis inferiores



< 1 KB	< 16 MB	< 16 GB	> 100 GB
0,25-0,5 ns	0,5-25 ns	80-250 ns	5 ms
20000-100000 MB/s	5000-10000 MB/s	1000-5000 MB/s	20-150 MB/s
CMOS	CMOS SRAM	CMOS DRAM	disco magnético
compilador	*hardware*	sistema operacional	sistema operacional e usuário

- O objetivo é manter os dados mais usados nas memórias mais rápidas, próximas à CPU

- Aumentar a velocidade da memória e diminuir o tempo de acesso são cruciais

Princípio da Localidade

- afirma que programas tendem a acessar uma pequena parte da memória em um dado momento
- Isso ocorre devido a repetições (iterações), sequências de instruções e estruturas de dados em bloco
- Este princípio é a base para a otimização da hierarquia de memória

Componentes da Localidade

- Localidade temporal
  - Dados usados recentemente provavelmente serão usados novamente
- Localidade espacial
  - Dados próximos aos usados recentemente provavelmente serão usados
- Localidade sequencial
  - O próximo dado na sequência provavelmente será usado

Taxas de Desempenho

- Taxa de acerto (hit ratio) “h”
  - Proporção de acessos à memória encontrados em um nível da hierarquia
- Taxa de falha (miss ratio) “m”
  - Proporção de acessos não encontrados em um nível da hierarquia, calculada por m = 1 - h
- Ciclos de parada por memória (ncp)
  - Número de ciclos que a CPU espera em caso de falha de acesso

Organização de memória

- A hierarquia de memória organiza a memória em níveis com diferentes velocidades e custos
- A memória principal é dividida em blocos de tamanho b, geralmente uma potência de 2
- Cada bloco possui um endereço, que é um múltiplo de b
- Um bloco é a menor unidade de informação transferida entre níveis

Desempenho da Memória

- O tempo de execução da CPU é afetado pelo número de ciclos de parada por memória

- ncp = ne × pe
  - Os ciclos de parada (ncp) dependem do número de erros de cache “misses” (ne) e da penalidade por erro “custo em ciclos” (pe)

- ncp = ni × (erros/instrução) × pe
  - O ncp também pode ser expresso em relação ao número de instruções (ni) e a frequência de erros por instrução

- ncp = ni × (acessos à memória/instrução) × m × pe
  - Detalha como o número de ciclos de parada da CPU é afetado pelo número de instruções, a frequência de acessos à memória, a taxa de erros de cache (m), e a penalidade por erro.

Operação do Cache

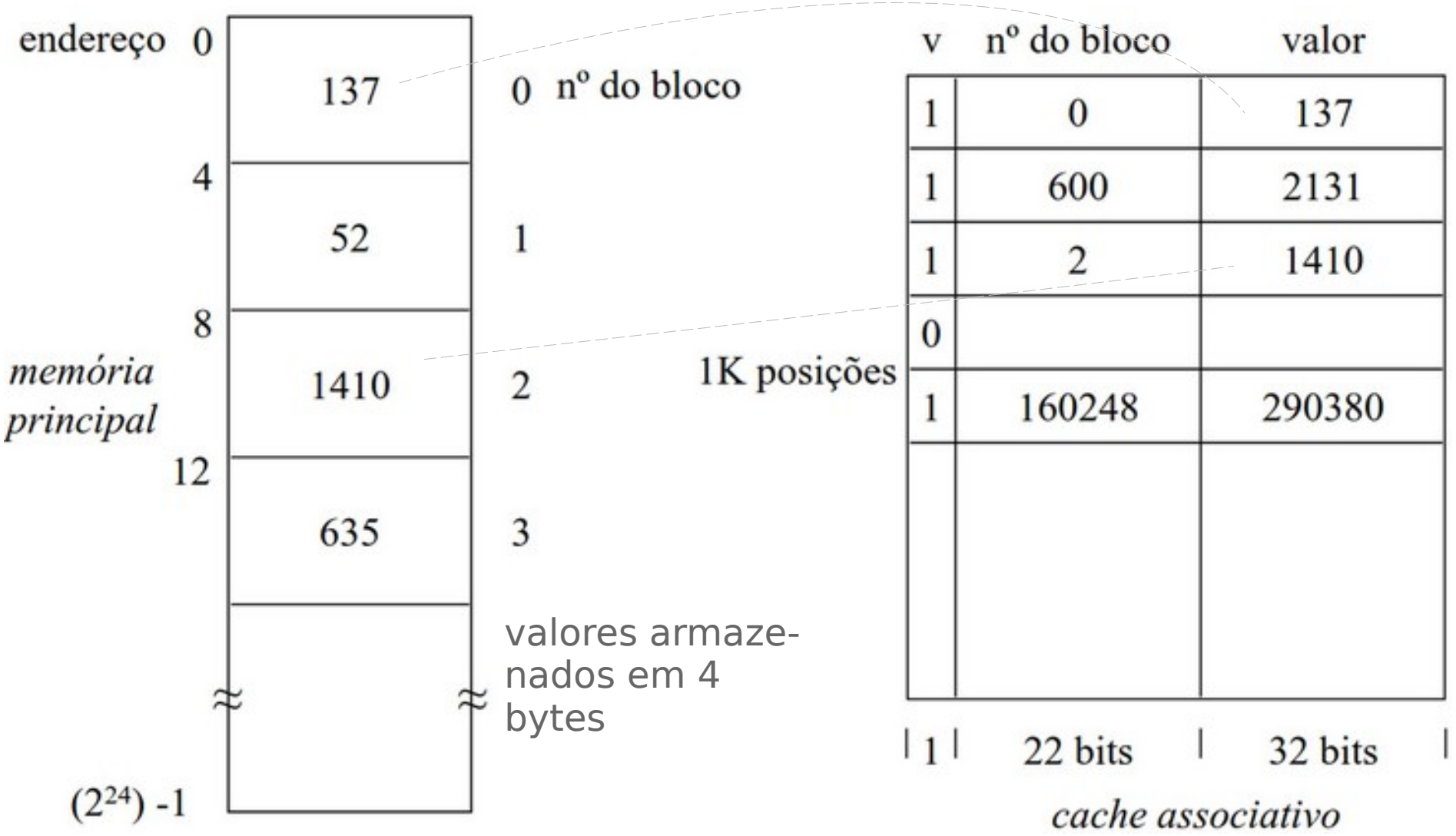
- O cache armazena blocos e seus números, junto com um bit que indica se a posição está em uso
- Quando a CPU precisa de um dado, o hardware calcula o número do bloco e procura no cache
- Se o bloco estiver no cache, o acesso é rápido; caso contrário, há uma falha (miss).
- Se o cache está cheio, um bloco é substituído para dar lugar a um novo

Tipos de Cache

- No cache associativo, a comparação é feita simultaneamente em todas as entradas, o que o torna mais caro
- No cache com mapeamento direto, cada bloco é alocado em uma posição específica, simplificando a busca
- A ordem das entradas no cache associativo é aleatória
- A escolha do tipo de cache influencia o desempenho e o custo do sistema

Cache associativo

No cache associativo, a busca por um bloco específico envolve verificar as etiquetas de todas as entradas do cache simultaneamente, o que aumenta a complexidade do circuito, mas melhora o desempenho geral.



http://orgcomp2.ic.uff.br/capitulo4/CacheAssociativo.php

Cache com mapeamento direto

posição	v	*tag*	valor	endereços
0	1	0	137	0, 4096, 8192, 12288, ...
1	1	600	2131	4, 4100, 8196, 12292, ...
2	1	2	1410	8, 4104, 8200, 12296, ...
3	0			
1023	0			4092, 8188, 12284, 16380, ...

Lista de endereços de memória que mapeiam para aquela posição na cache.
O mapeamento é feito usando uma função que geralmente envolve o módulo (resto da divisão) do número do bloco pelo número de linhas da cache.

- Se ocache estiver cheio, uma entrada terá que ser descartada para deixar lugar para uma nova

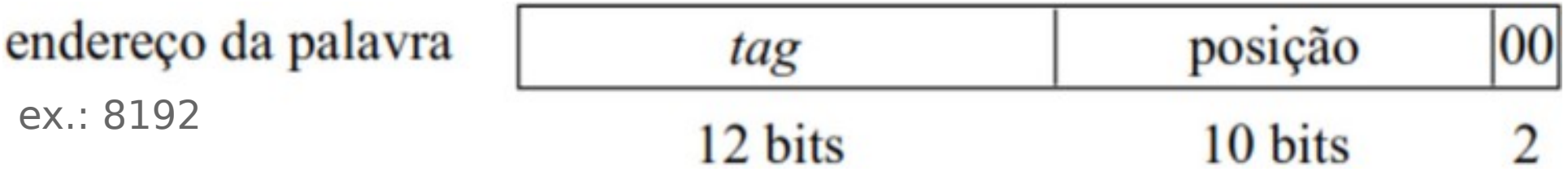
- Quando aparece um endereço de memória, o microprograma deve calcular o número do bloco e, então, procurar aquele número no cache

- Para evitar a pesquisa linear, o cache associativo pode fazer uso de uma memória associativa, que compara simultaneamente todas as entradas com o número do bloco dado. Isto torna o cache associativo caro

- No cache com mapeamento direto, cada bloco é colocado numa posição, cujo número pode corresponder, por exemplo, ao resto da divisão do número do bloco pelo número de posições

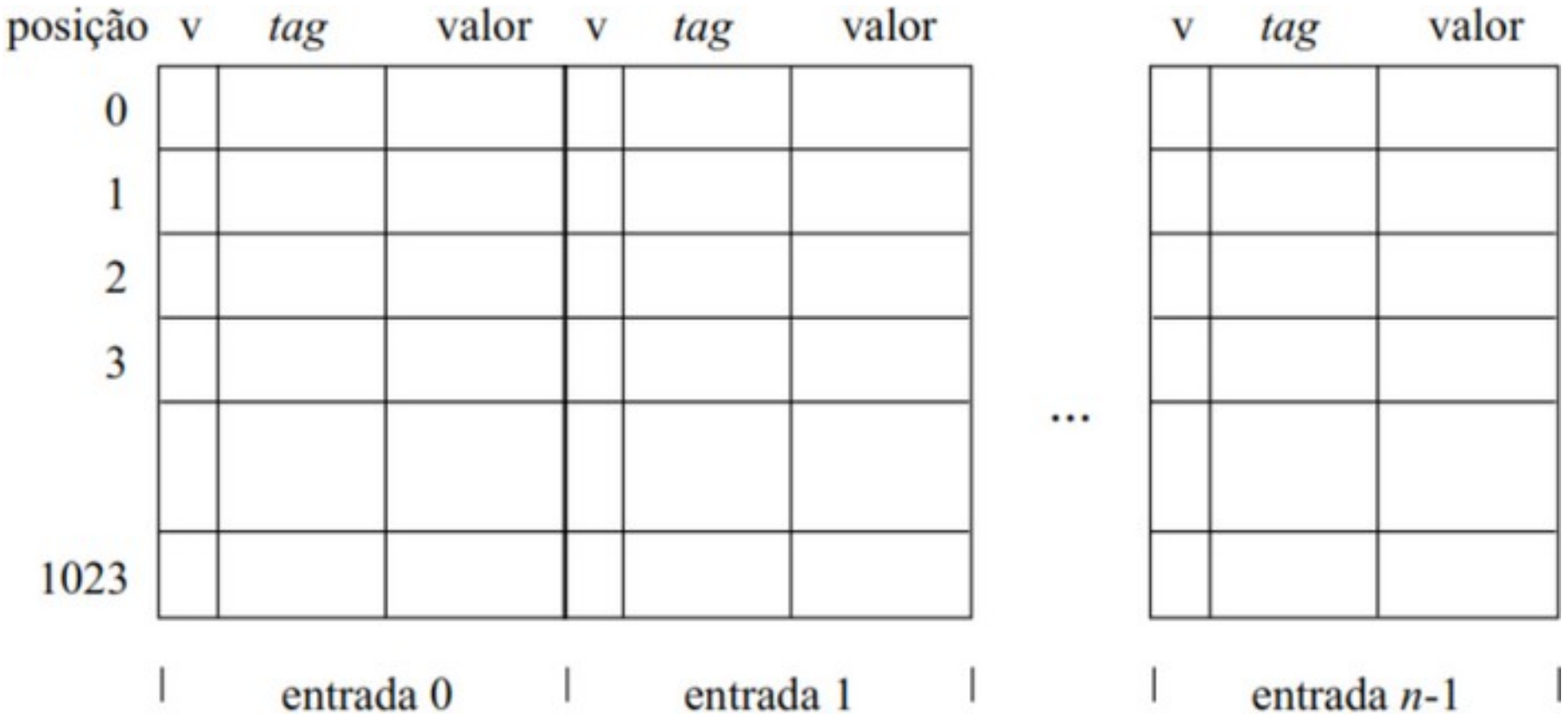
Endereço de uma palavra na cache

Como encontrar um byte específico



- Tag (12 bits)
  - É a parte do endereço que identifica um bloco na memória
  - Ele não participa diretamente do endereçamento da posição na cache, mas é usado para verificar se o bloco correto está armazenado na posição da cache
- Posição (10 bits) ($2^{10} = 1024$)
  - Identifica a linha específica na cache onde o bloco pode ser armazenado
  - É usado para indexar diretamente na cache
- Dois Bits Menos Significativos
  - Indicam a posição exata dentro do bloco (os blocos têm 4 bytes)
  - São usados para acessar a palavra específica dentro do bloco

Cache de mapeamento direto



- No cache associativo por conjunto utiliza-se um cache de mapeamento direto com múltiplas entradas por posição

- Tanto o cache associativo quanto o de mapeamento direto são casos especiais do cache associativo por conjunto

- O cache de mapeamento direto é mais simples, mais barato e tem tempo de acesso mais rápido

- O cache associativo tem uma taxa de acerto maior para qualquer dado número de posições, pois a probabilidade de conflitos é mínima

Técnica de escrita/leitura

- Write through
  - A escrita é feita tanto no cache quanto na memória principal simultaneamente. Isso garante consistência imediata dos dados, mas gera mais tráfego no barramento
  - É mais adequada quando há muitas leituras e poucas escritas
- Copy back
  - A escrita é feita apenas no cache
  - A memória principal é atualizada somente quando o bloco é removido do cache
  - Essa técnica pode causar inconsistência se houver transferência de dados entre memória e disco antes da atualização
  - É preferível quando há muitas escritas, com o microprograma limpando o cache antes de operações de entrada e saída