Introdução à Informática

Alexandre Meslin
(meslin@nce.ufrj.br)

cederj

Organização da Memória

- Conceito de hierarquia de memória
- Memória principal e memórias secundárias
- Projeto lógico da memória principal
- Memórias cache
- Memória virtual

Conceito de Hierarquia de Memória

- Memória no computador
  - **Registradores**
  - **Memória principal**
  - **Memória cache**
  - Memória ROM
  - Discos magnéticos
  - Discos ópticos
  - Fitas magnéticas

Registradores

- É a memória mais rápida
- Montada dentro da CPU
- Local onde as operações aritméticas são calculadas
- Conjuntos de apenas algumas dezenas

Memória Principal

- Construída utilizando memória dinâmica
- Memória de leitura e escrita
- DRAM
- Mais lenta e mais barata que a memória cache
- Computadores atuais possuem pelo menos 64 megabytes
- Tempo de acesso unitário de 60 ns e em modo rajada de 7 ns
- O seu conteúdo é perdido quando a energia é desligada

Problemas

- Configuração de hardware
  - Processador de 1 GHz
  - Unidade aritmética de 32 bits (4 bytes)
  - Tempo de acesso à memória:
    - Unitário: 60 ns
    - Rajada: 7 ns (PC133)
- Processador necessita de dados a cada 1 ns
  - Período é o inverso da freqüência
  - 1 ns = 1/1 GHz

Valores

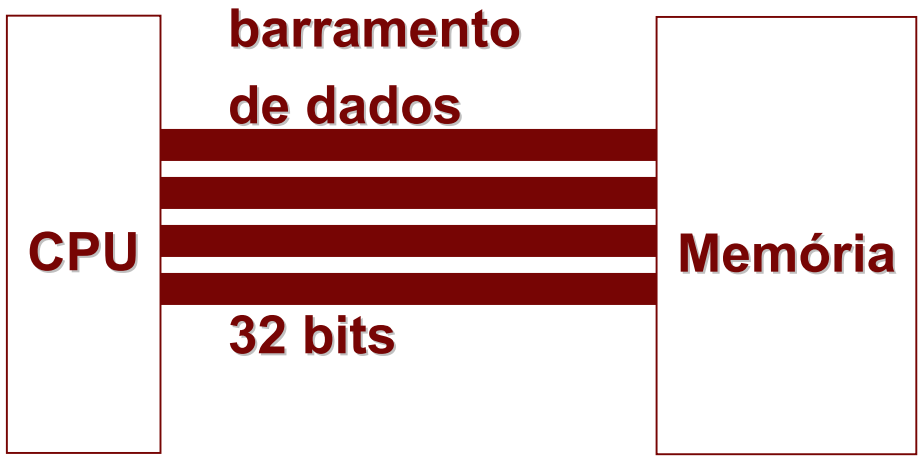
- Memória de 8 bits
  - 4 acessos necessários
  - 1o acesso em 60 ns
  - 2o, 3o e 4o acesso a cada 7 ns
  - Tempo total = 60 + 3*7 = 81 ns
  - Memória 80 vezes mais lenta que o processador

Valores

- Memória de 16 bits
  - 2 acessos necessários
  - 1o acesso em 60 ns
  - 2o acesso em 7 ns
  - Tempo total = 60 + 7 = 67 ns
  - Memória 67 vezes mais lenta que o processador

Valores

- Memória de 32 bits
  - 1 único acesso
  - acesso em 60 ns
  - Tempo total = 60 ns
  - Memória 60 vezes mais lenta que o processador



Memória Cache

- Construída utilizando memória estática
- Memória de leitura e escrita
- SRAM
- Extremamente rápida
- Pouca capacidade
- Opera em velocidade perto da velocidade do processador
  - Igual ou metade
- Memória de acesso aleatório

Memória Cache

- Nível 1 → construída junto ao processador
- Nível 2 → fora do processador (na placa mãe)
- A maior parte dos PC's contém:
  - Alguns quilobytes de nível 1
  - Poucos megabytes de nível 2
- O tempo de acesso é menor do que 6ns
- Memória muito cara
- O seu conteúdo é perdido quando a energia é desligada

Termos Comuns

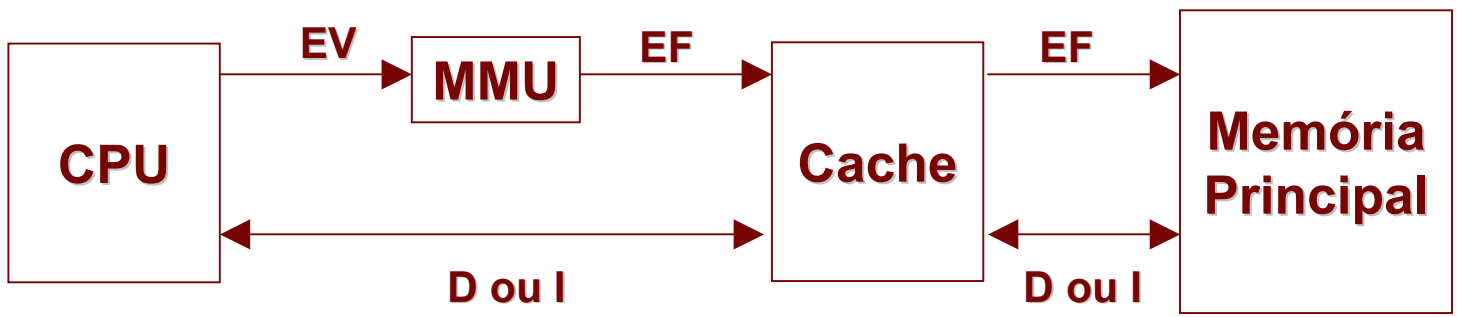
- Referências pelo processador a dados armazenados em cache são chamados de ACERTO (HIT)
  - Taxa de acerto normalmente maior que 95%
- Referências pelo processador a dados não armazenados em cache são chamados de FALHA (MISS)
- Cache normalmente busca uma linha
  - Localidade espacial
- Descritor (tag)
  - Dados, endereços, validade, etc.

Funcionamento

- Tentativa de aproximar o tempo de acesso à memória do tempo de acesso da CPU
- Dependente de diversos fatores
  - Arquitetura do computador
  - Comportamento dos programas
  - Tamanho e organização da cache
- Transparente para o programa/programador

Funcionamento

- Pedido de memória verificado antes na cache
  - Se estiver presente, a cache fornece/recebe informação da CPU (acerto ou hit)
  - Caso contrário, o pedido é enviado para a memória principal (falha ou miss)



Princípios da Cache

- Localidade espacial
  - ❖ Grande probabilidade de acessos à locais vizinhos
- Localidade temporal
  - ❖ Grande probabilidade de se acessar regiões que foram recentemente acessadas
- Seqüencialidade
  - ❖ Se uma referência é feita ao endereço X, existe grande probabilidade de haver referência ao endereço X+1

Memória Cache

Organização do Cache

- Mapeamento direto
  - Direct mapped
- Conjunto associativo
  - Set associative
- Totalmente associativo
  - Fully associative

Curiosidades

- Supondo memória cache com:
  - 256 kbytes
  - 32 bytes/bloco
  - 8 linhas
  - Barramento de endereço de 32 bits
  - Capacidade de endereçar até 4 Gbytes
  - Mapeamento direto

Cache com Mapeamento Direto

- Curiosidades
  - Os bytes 0, 1, 2 até o byte 31 estão na primeira linha da cache
  - Os bytes 32, 33, 34 até o byte 63 estão na segunda linha da cache
  - Os bytes 64, 65, 66 até o byte 127 estão na terceira linha da cache

Cache com Mapeamento Direto

- Convertendo os números para binário, observa-se:
  - Os 5 bits menos significativos dos elementos que pertencem à mesma linha são diferentes
  - Os outros bits são todos iguais.

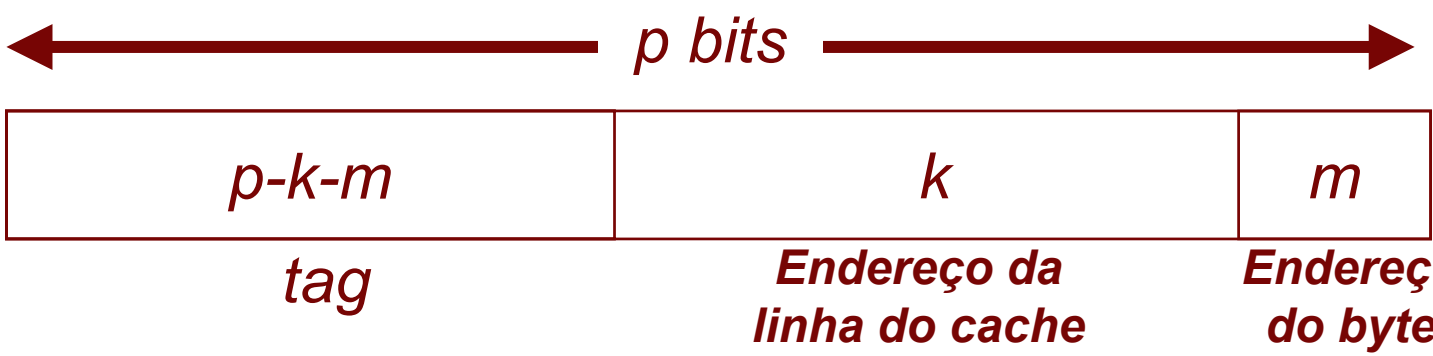
Cache com Mapeamento Direto

- Como o cache tem 256 kbytes endereços, convertendo para binário, isto representa um número de 18 bits

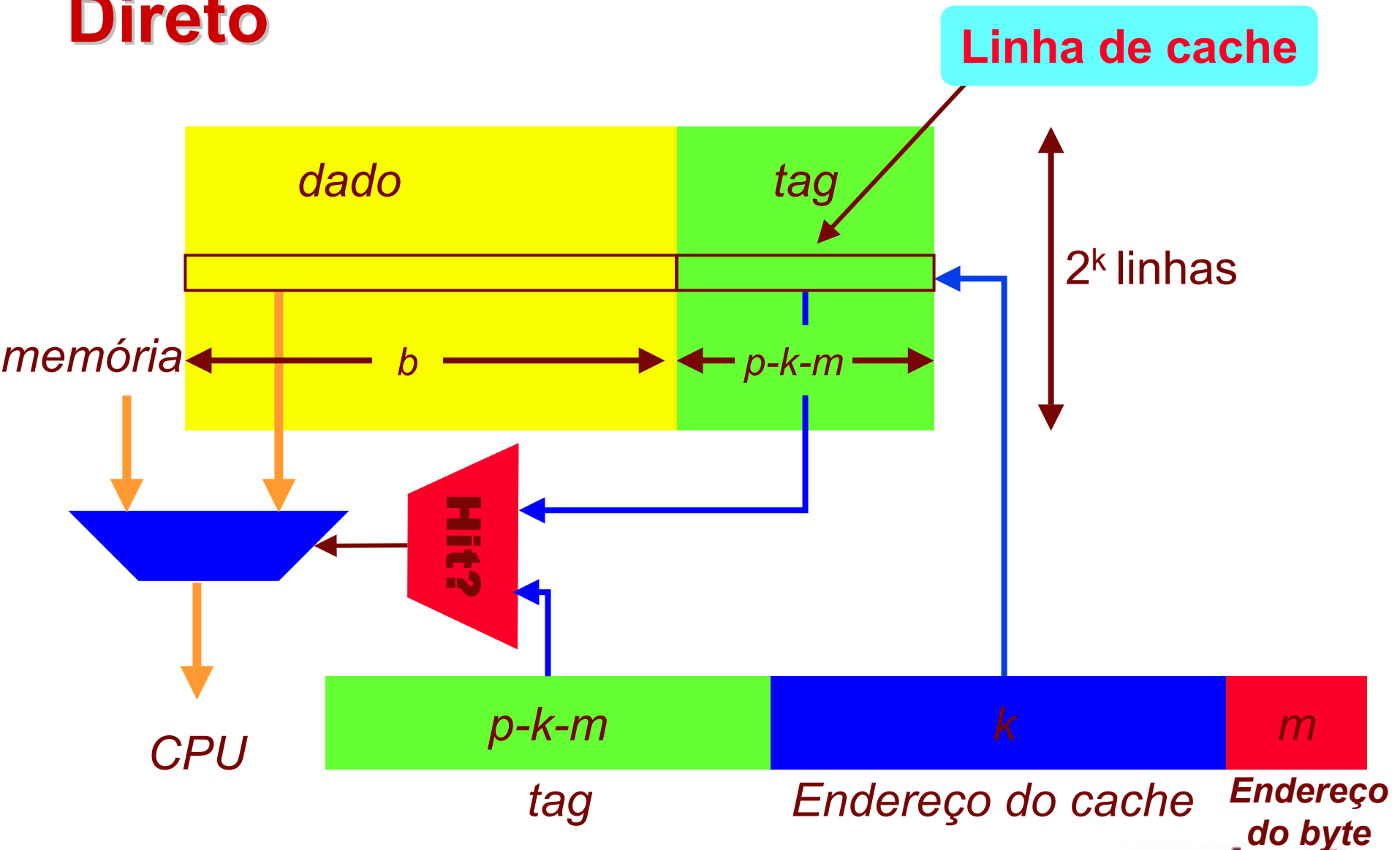
- 2^{18} = 262144 = 256 kbytes

Cache com Mapeamento Direto

- Assumindo que:
  - Memória cache tem 2^k linhas
  - Memória cache tem bloco com 2^m bytes
  - p bits de barramento de endereço
- Bits mais baixos utilizados para selecionar a linha da cache
- Bits mais altos usados para comparar o endereço da linha



Cache com Mapeamento Direto



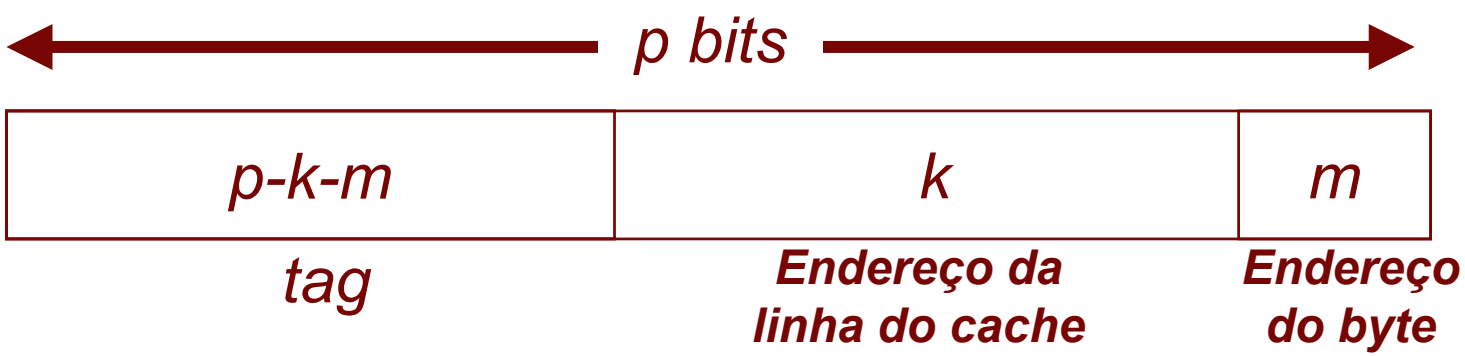
Exemplo

- Supondo memória cache com:
  - 256 kbytes
  - 32 bytes/bloco
  - 8 k linhas
  - Barramento de endereço de 32 bits
  - Mapeamento direto
- Calcular o bloco que será utilizado pelo endereço:

87a6c1b4 (hexadecimal)

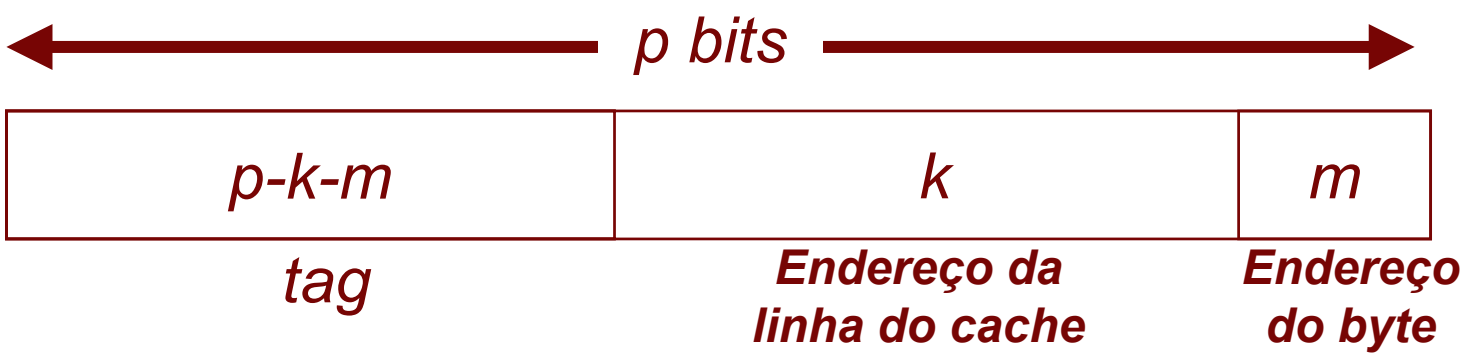
Exemplo

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 8 7 a 6 c 1 b 4
  - 1000 0111 1010 0110 1100 0001 1001 0100



Exemplo

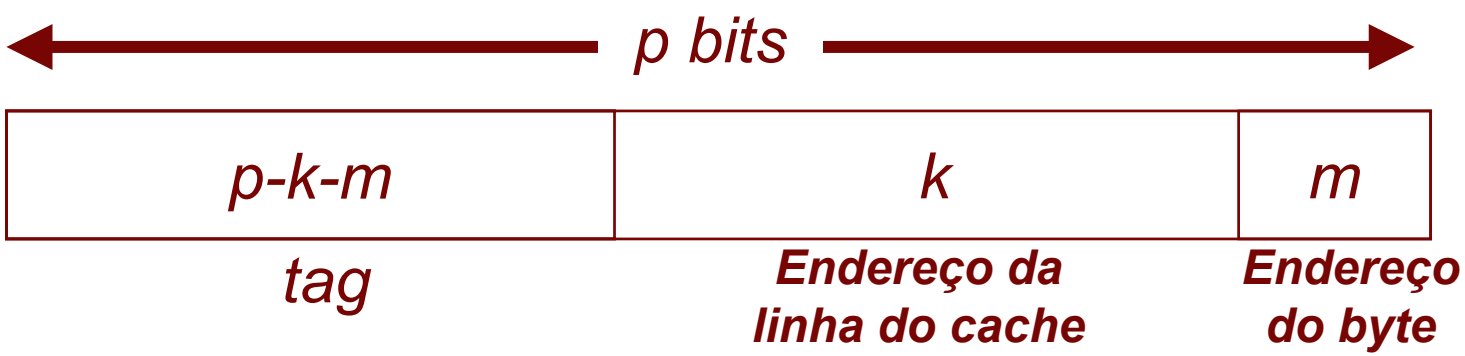
- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 8 7 a 6 c 1 b 4
  - 1000 0111 1010 0110 1100 0001 10010100



cederj

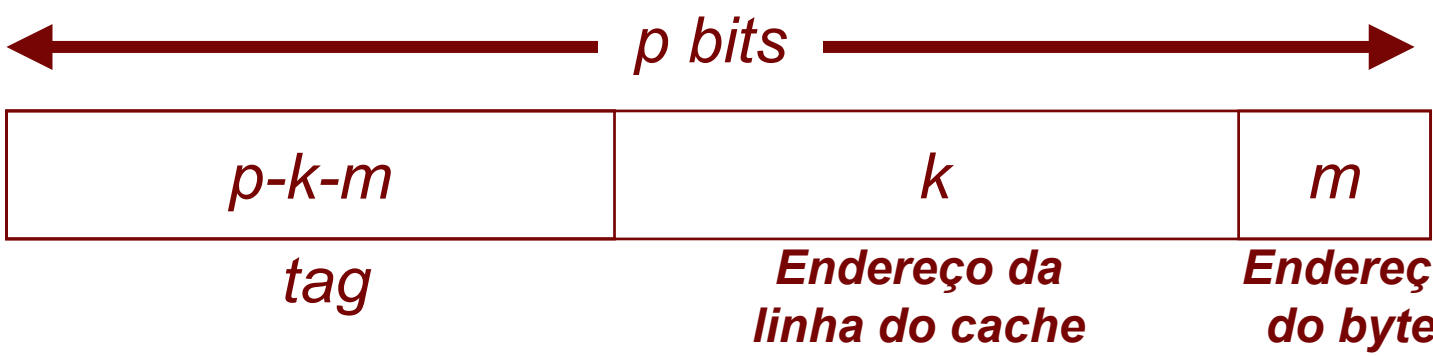
Exemplo

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 8 7 a 6 c 1 b 4
  - ❖1000 0111 1010 0110 1100 000110010100



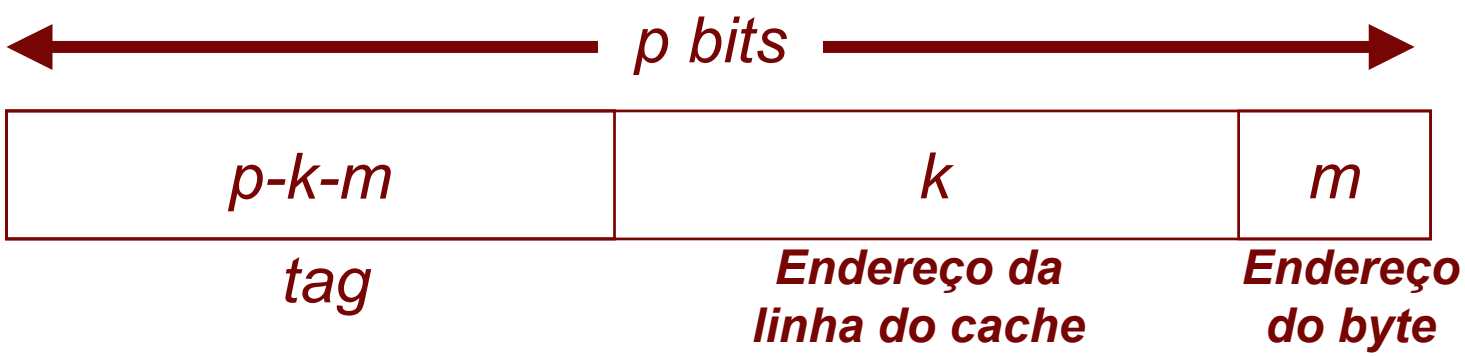
Exemplo

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 8 7 a 6 c 1 b 4
  - 1000 0111 1010 0110 1100000110010100



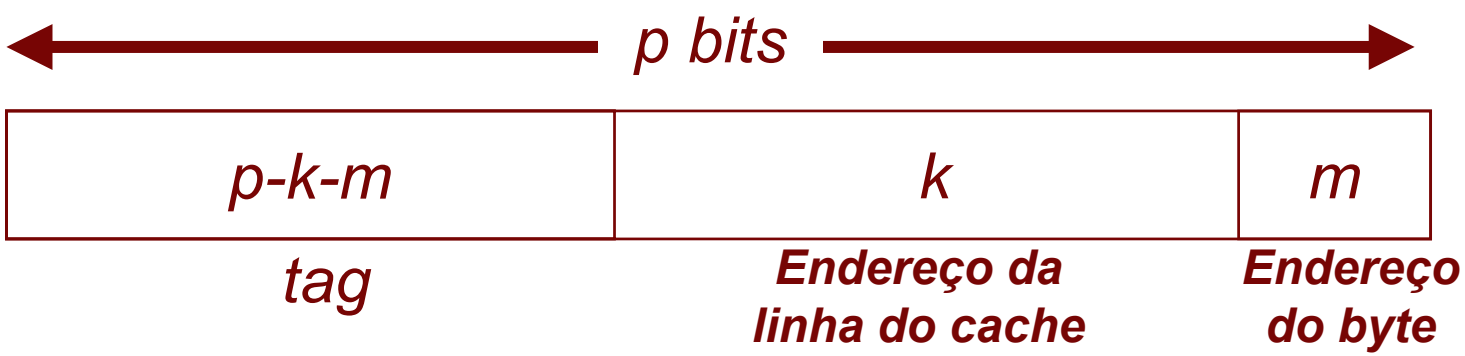
Exemplo

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 8 7 a 6 c 1 b 4
  - 1000 0111 1010 01101100000110010100



Exemplo

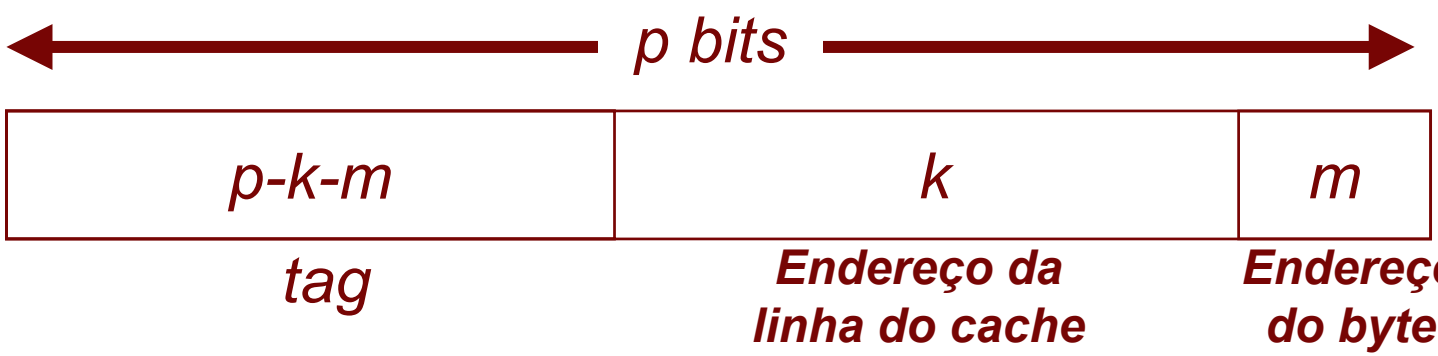
- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 8 7 a 6 c 1 b 4
  - 1000 0111 10100110110000110010100



cederj

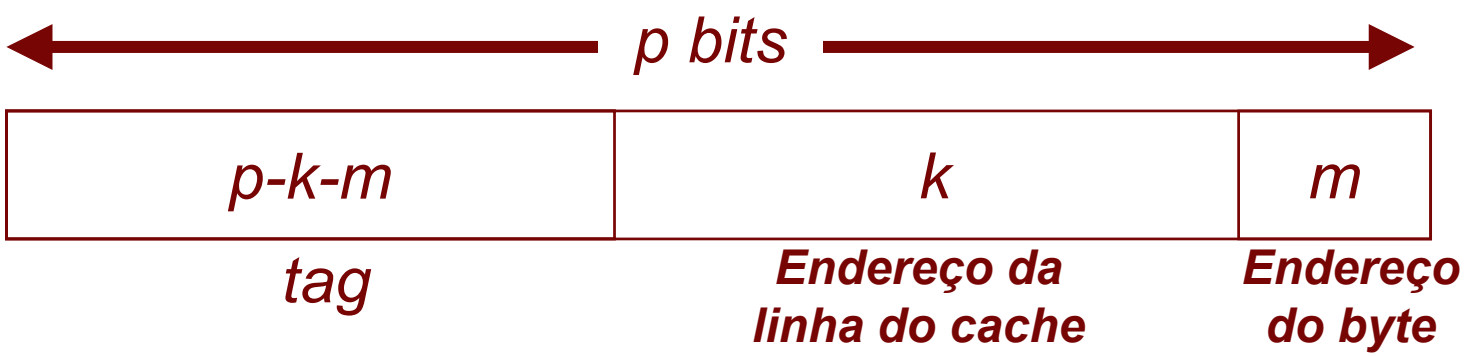
Exemplo

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 8 7 a 6 c 1 b 4
  - 1000 01111010011011000001100101000



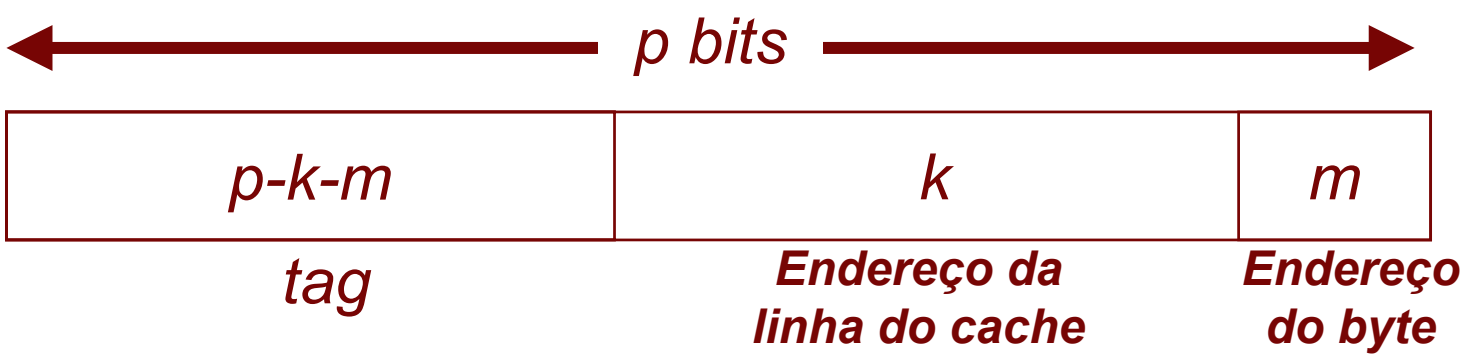
Exemplo

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 8 7 a 6 c 1 b 4
  - 10000111101001101100001100101100



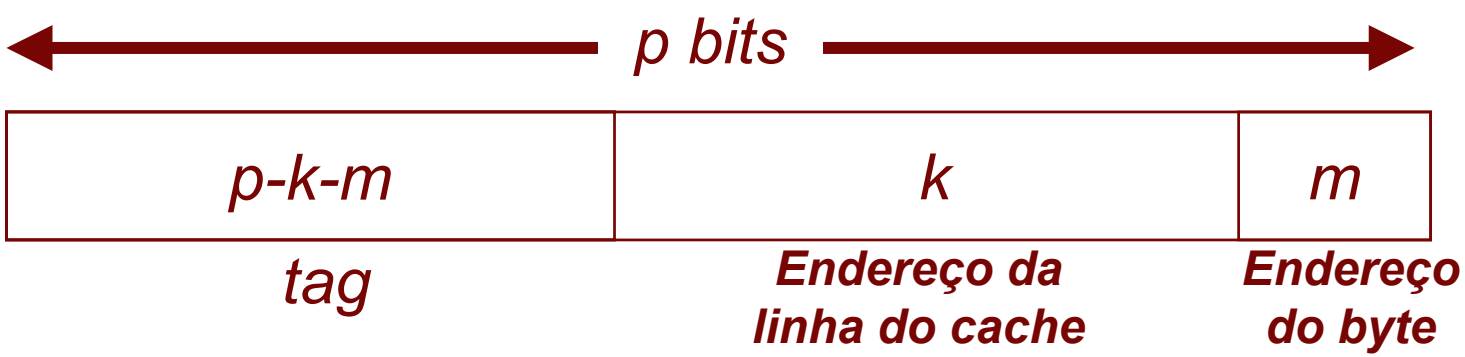
Exemplo

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 87a6c1b4
  - 10000111101001101100000110010100



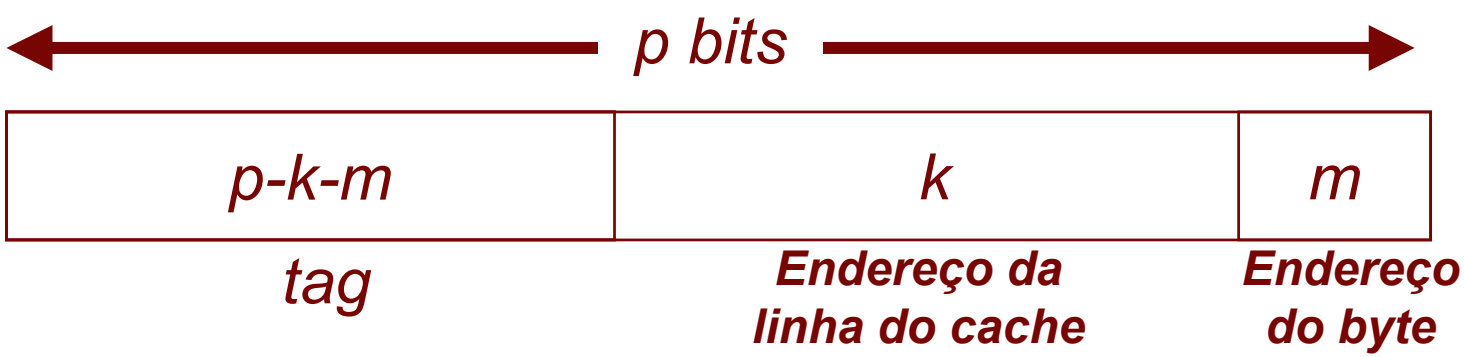
Exemplo

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 87a6c1b4
  - ❖10000111101001101100001100 10100



Exemplo

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 87a6c1b4
  - `10000111101001 1011000001100 10100`
  - `k = 1011000001100`



Exemplo

- Bloco endereçado por 87a6c1b4 (hexadecimal)
- Linha de cache 10110000001100 (binário)
  - 160C (hexadecimal)
  - 5644 (decimal)

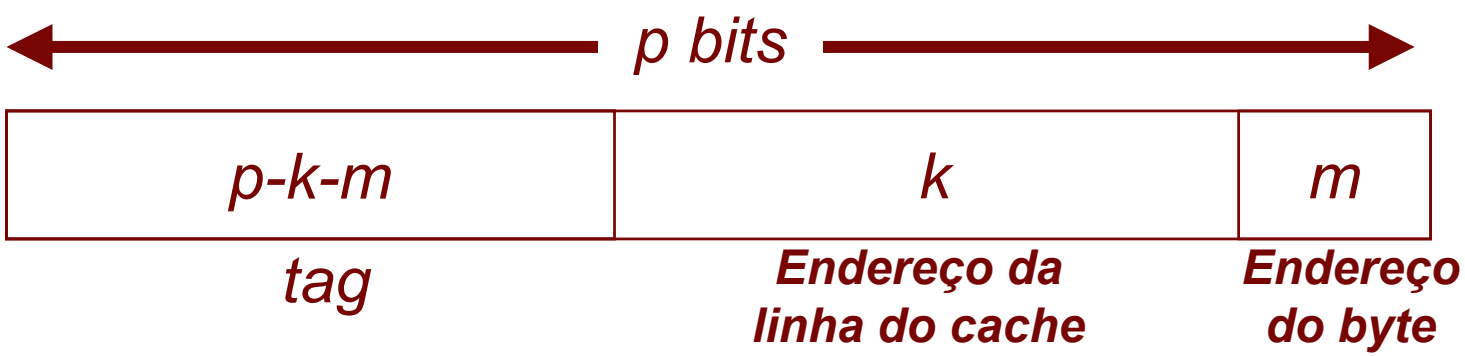
Problema

- Refazer o problema para o endereço

  88a6c1b4 (hexadecimal)

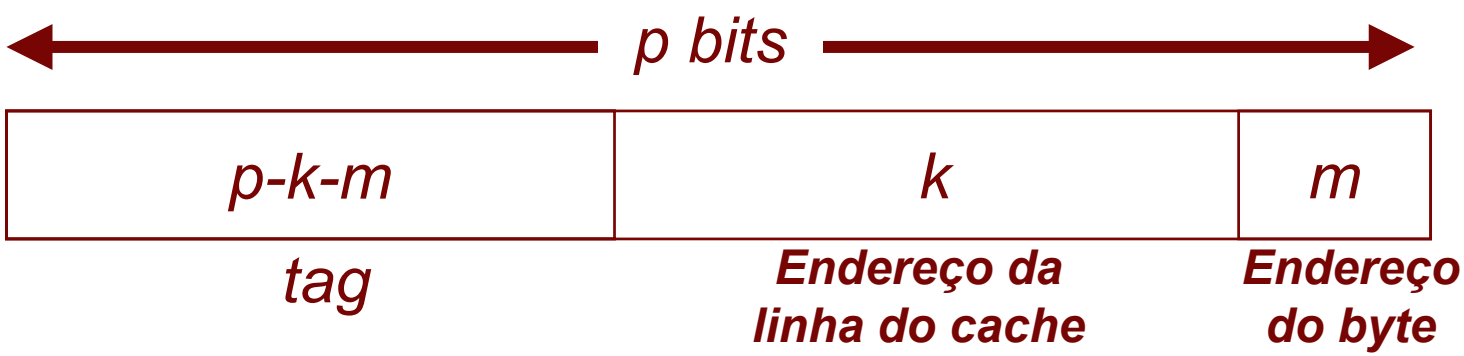
Problema

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 8 8 a 6 c 1 b 4
  - 1000 1000 1010 0110 1100 0001 1001 0100



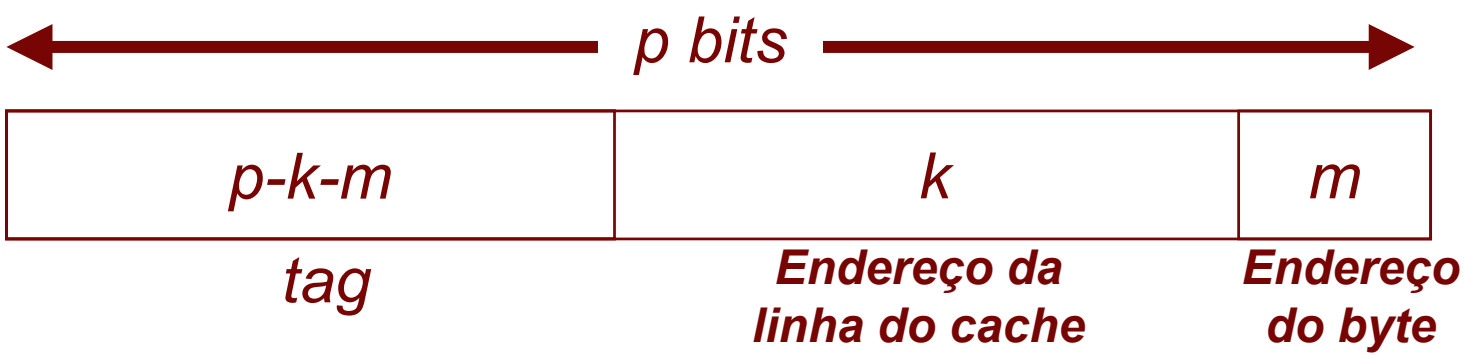
Problema

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 8 8 a 6 c 1 b 4
  - 1000 1000 1010 0110 1100 0001 10010100



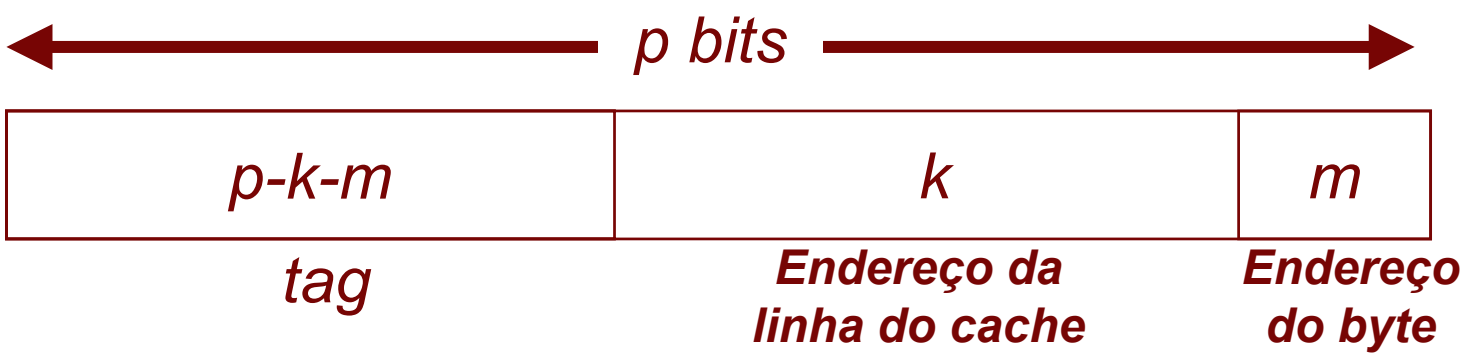
Problema

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 8 8 a 6 c 1 b 4
  - 1000 1000 1010 0110 1100 000110010100



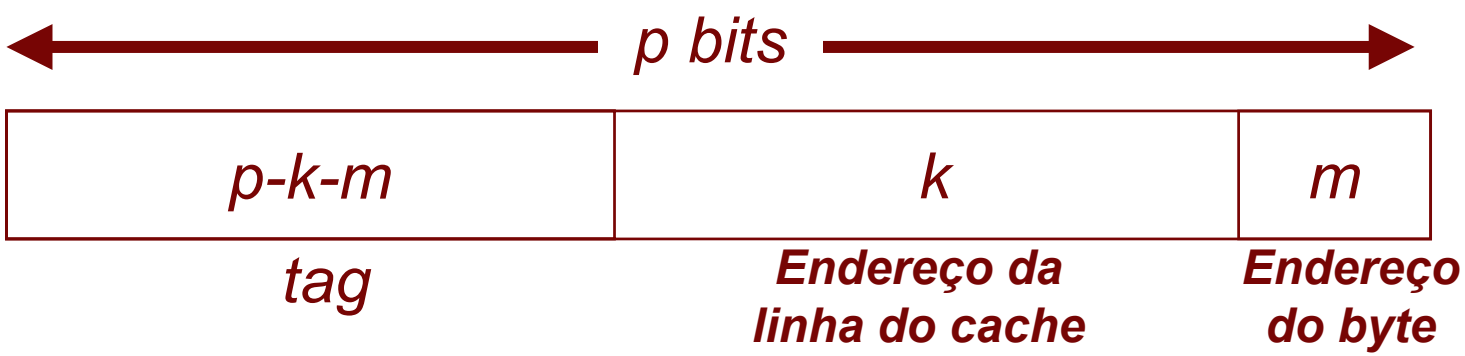
Problema

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 8 8 a 6 c 1 b 4
  - 1000 1000 1010 0110 11000001100101 00



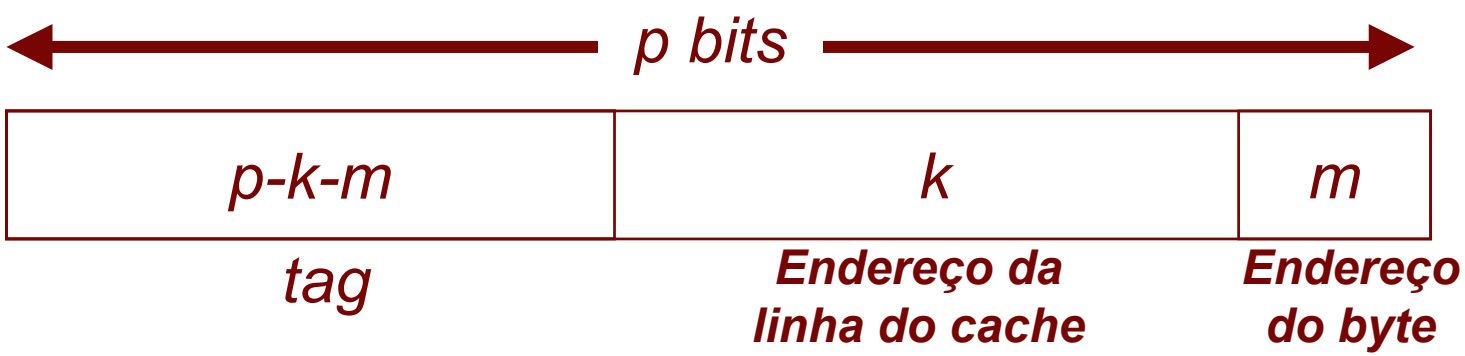
Problema

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 8 8 a 6 c 1 b 4
  - 1000 1000 1010 01101100000110010100



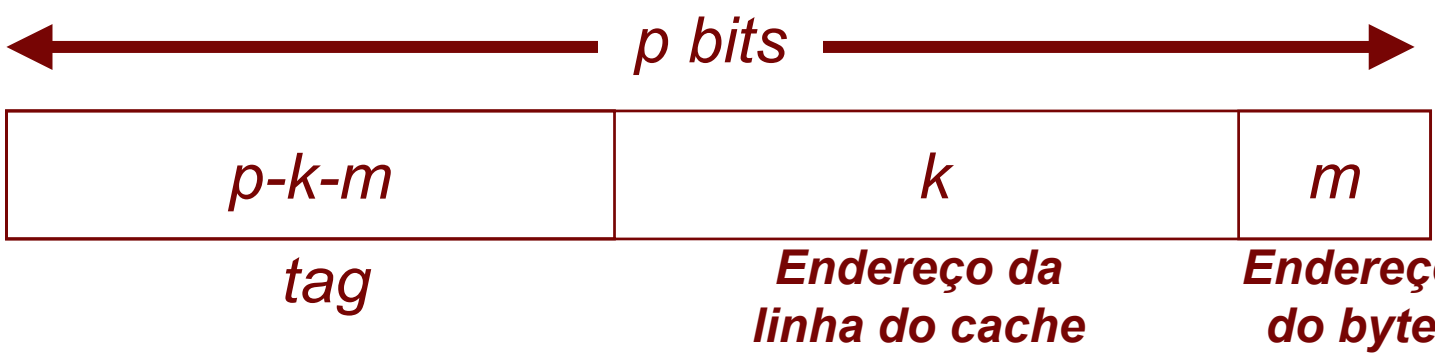
Problema

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 8 8 a 6 c 1 b 4
  - 1000 1000 10100110110000110010100



Problema

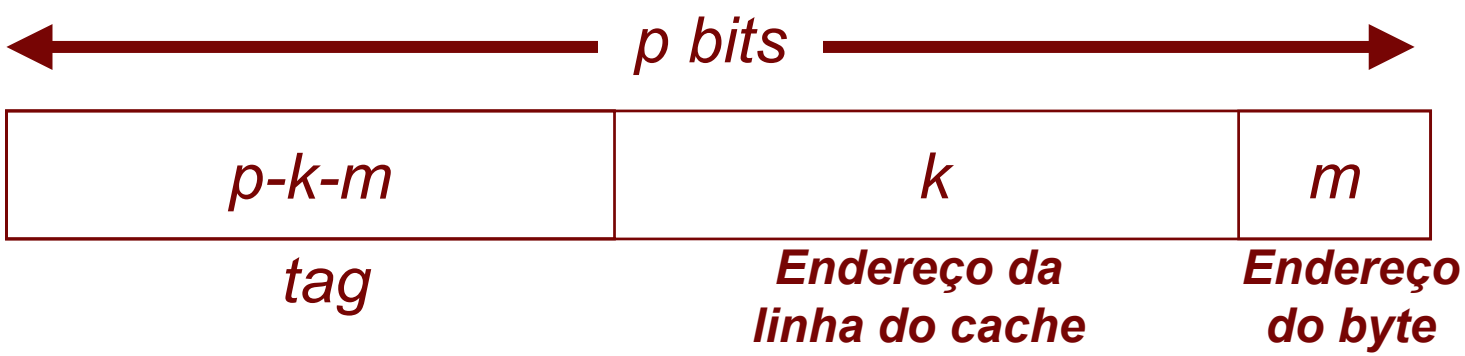
- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 8 8 a 6 c 1 b 4
  - 1000 10001010011011000001100010100



cederj

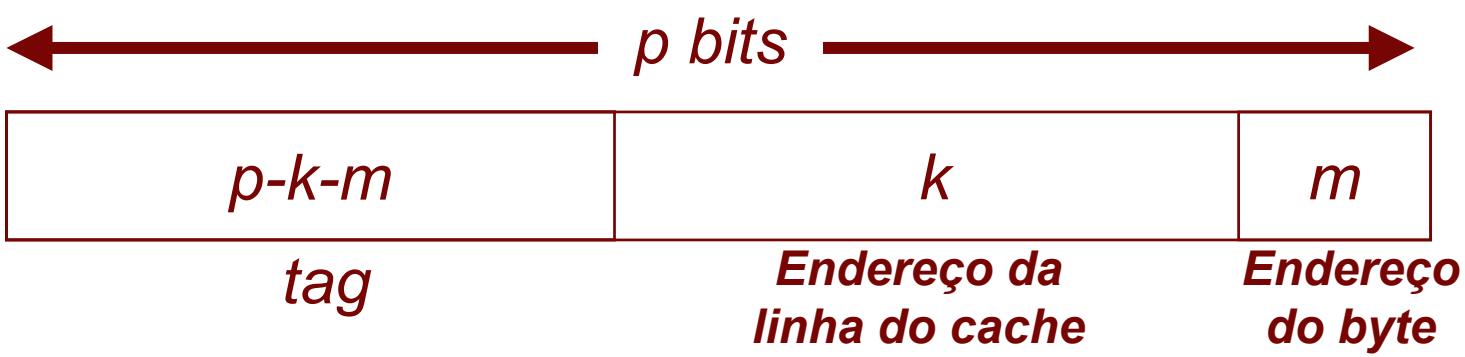
Problema

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 8 8 a 6 c 1 b 4
  - 10001000101001101100001100101100



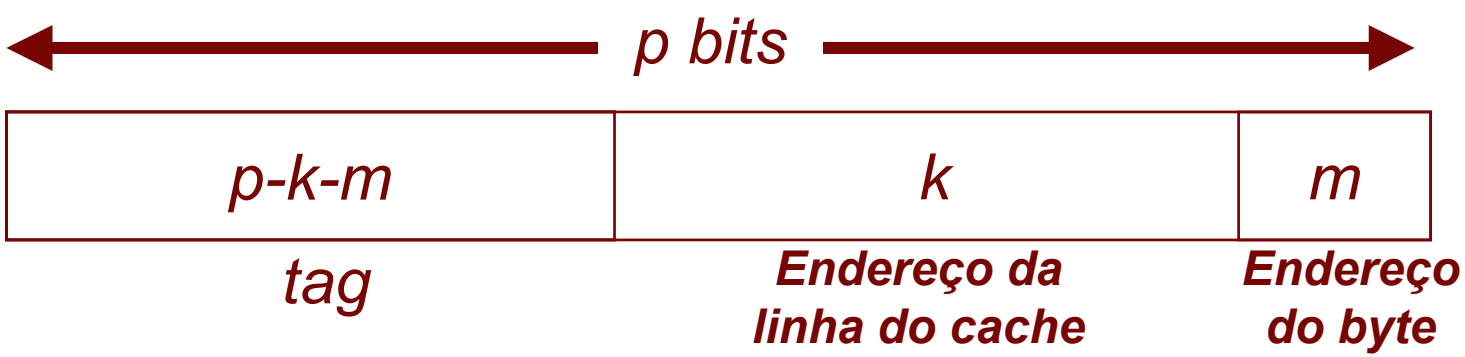
Problema

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 88a6c1b4
  - 10001000101001101100000110010100



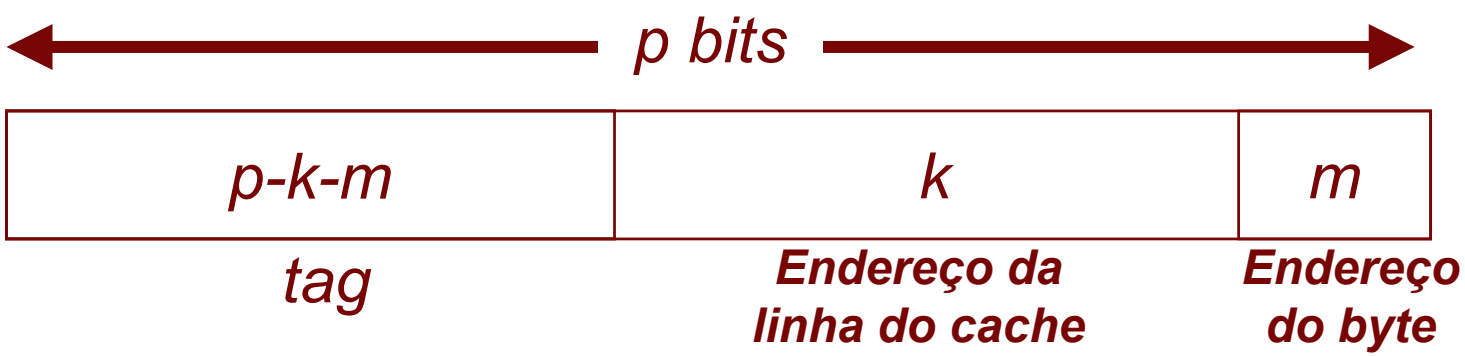
Problema

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 88a6c1b4
  - 10001000101001101100000 01100 10100



Problema

- p = 32 bits
- m = 5 bits ($2^5 = 32$)
- k = 13 bits (2^{13}=8 k)
- 88a6c1b4
  - `10001000101001 1011000001100 10100`
  - `k = 1011000001100`



Problema

- Bloco endereçado por 88a6c1b4 (hexadecimal)
- Linha de cache 10110000001100 (binário)
  - 160C (hexadecimal)
  - 5644 (decimal)

Problema

- Os endereços 87a6c1b4 e 88a6c1b4 necessitam da mesma linha de cache

- Conclusão:
  - Todo par de endereços cuja diferença for múltipla do tamanho da cache vai utilizar a mesma linha gerando conflito
  - Sempre que houver conflito, a linha mais antiga será descartada da cache

Próxima Aula

- Continuação de hierarquia de memória
- Tentativa de solucionar o problema de conflito no cache