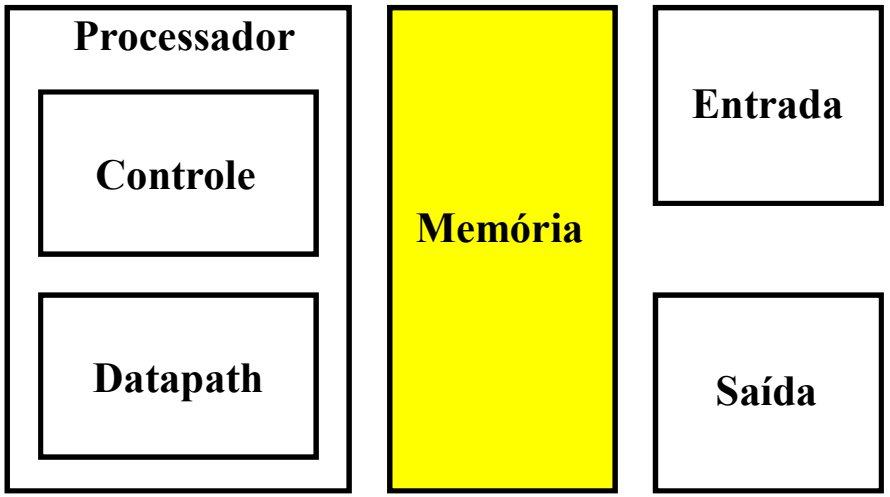
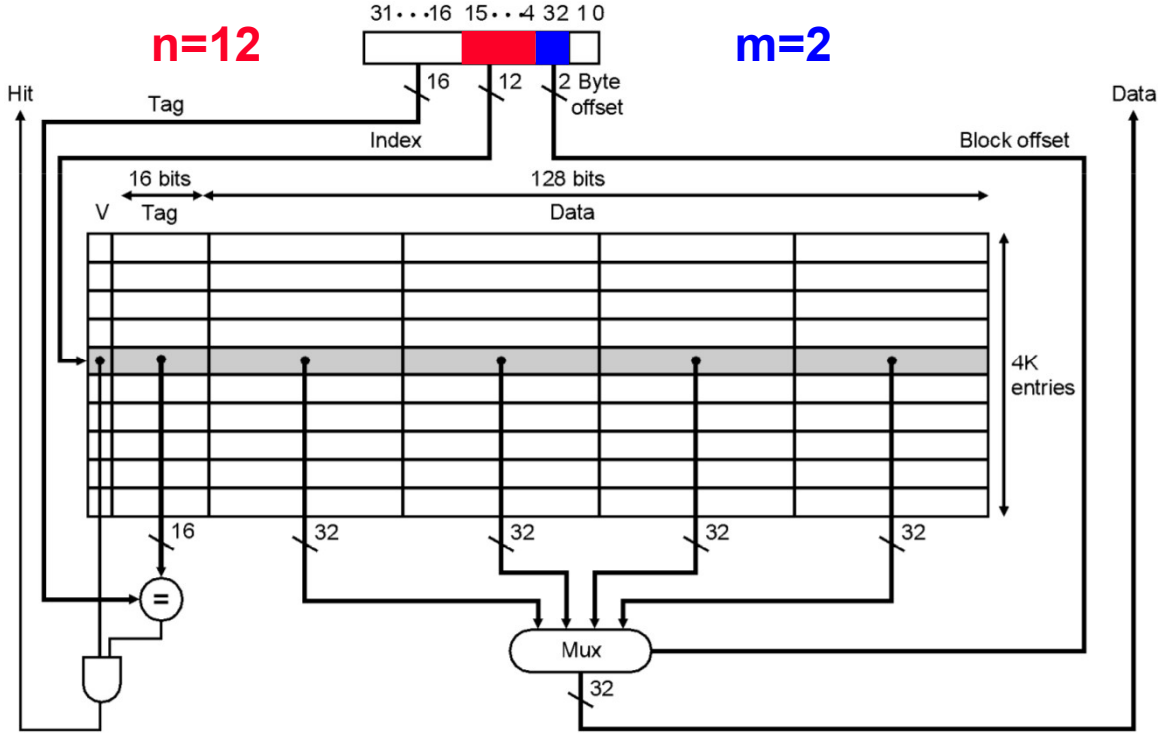
Cache: mapeamento e consistência com memória principal



Explorando a Localidade Espacial

- **Cache com m=0 só explora localidade temporal**
- **Idéia-chave: bloco > 1 palavra**
  - **Falta: busca de palavras adjacentes**



Mapeamento:

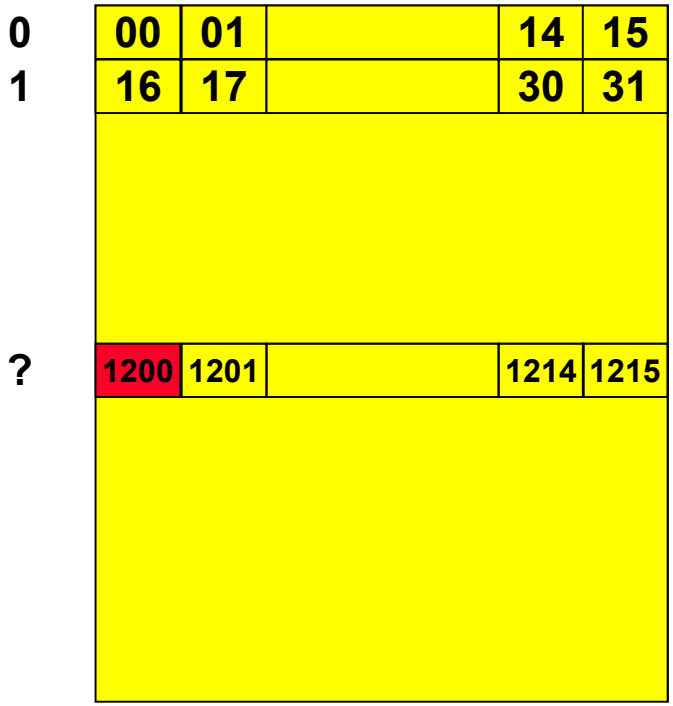
 # bloco modulo N,

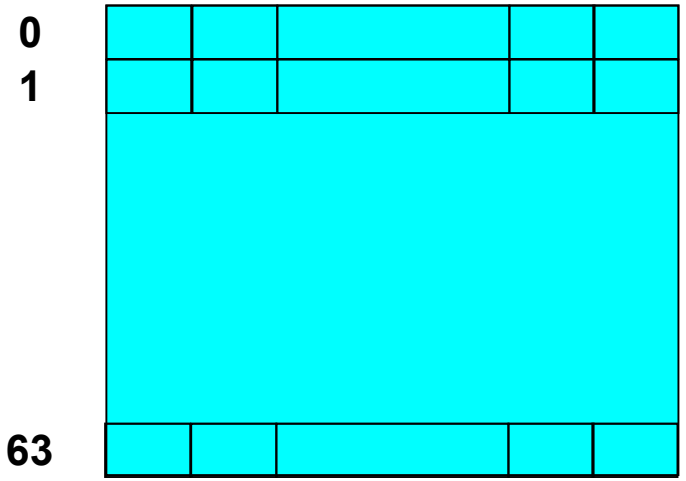
(onde N = 2^n)

$$\#\,\text{bloco} = \frac{\text{endereço do byte}}{\text{bytes por bloco}}$$

Múltiplas Palavras por Bloco

- **Exemplo de mapeamento**
  - **Cache armazena 64 blocos; bloco = 16 bytes.**
  - **Para que bloco da cache é mapeado o byte 1200 ?**

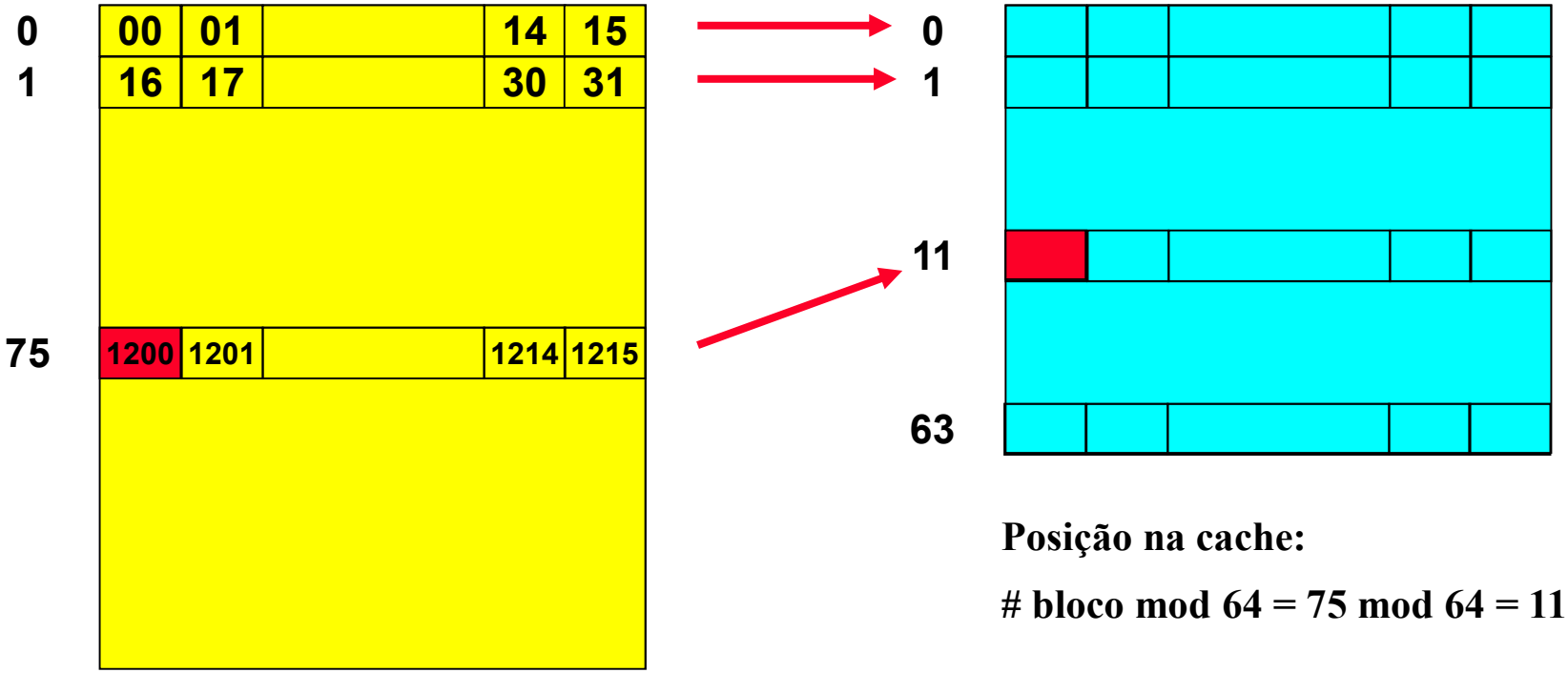




$$\#\text{bloco} = \frac{\text{endereço do byte}}{\text{bytes por bloco}} = \frac{1200}{16} = 75$$

Múltiplas Palavras por Bloco

- **Exemplo de mapeamento**
  - **Cache armazena 64 blocos; bloco = 16 bytes.**
  - **Para que bloco da cache é mapeado o byte 1200 ?**



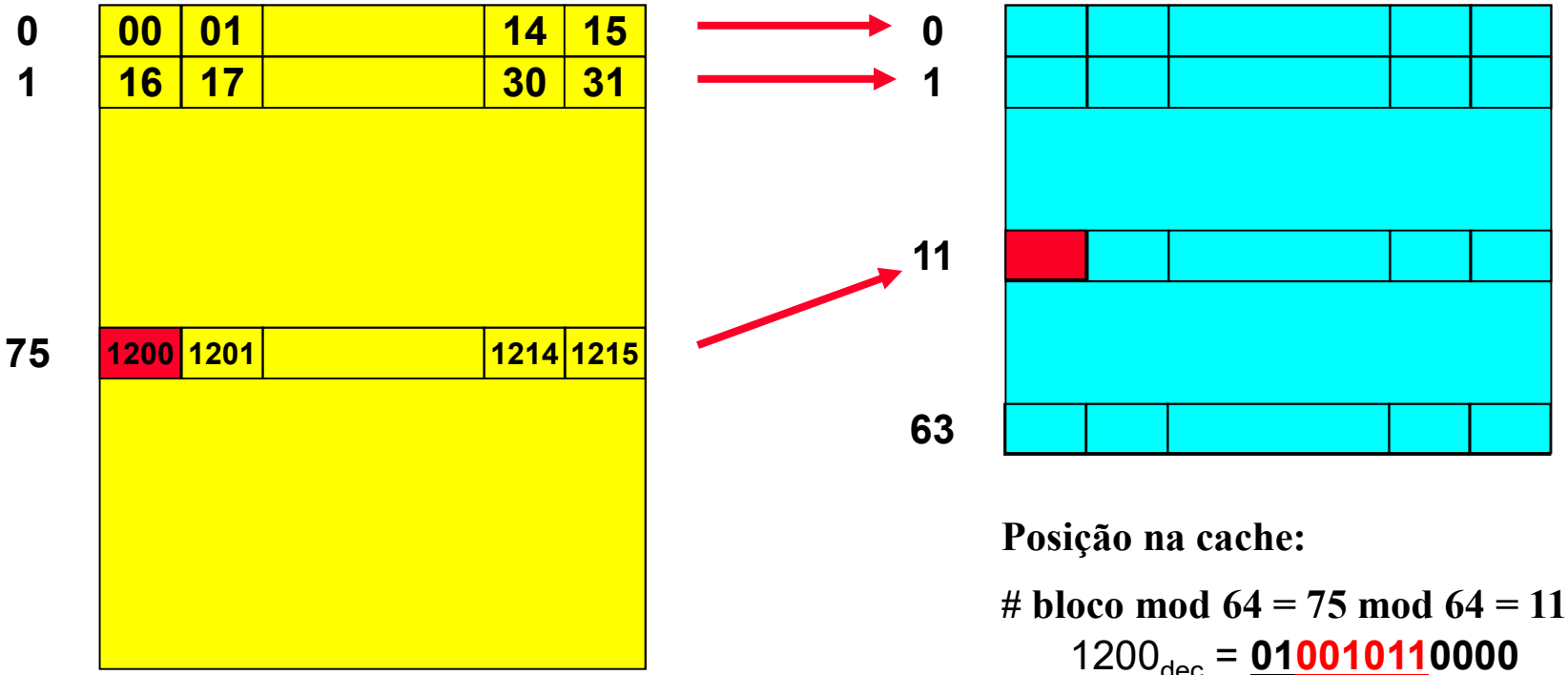
Posição na cache:

 # bloco mod 64 = 75 mod 64 = 11

$$\#\text{bloco} = \frac{\text{endereço do byte}}{\text{bytes por bloco}} = \frac{1200}{16} = 75$$

Múltiplas Palavras por Bloco

- **Exemplo de mapeamento**
  - **Cache armazena 64 blocos; bloco = 16 bytes.**
  - **Para que bloco da cache é mapeado o byte 1200 ?**



$$\#\text{bloco} = \frac{\text{endereço do byte}}{\text{bytes por bloco}} = \frac{1200}{16} = 75$$

Posição na cache:

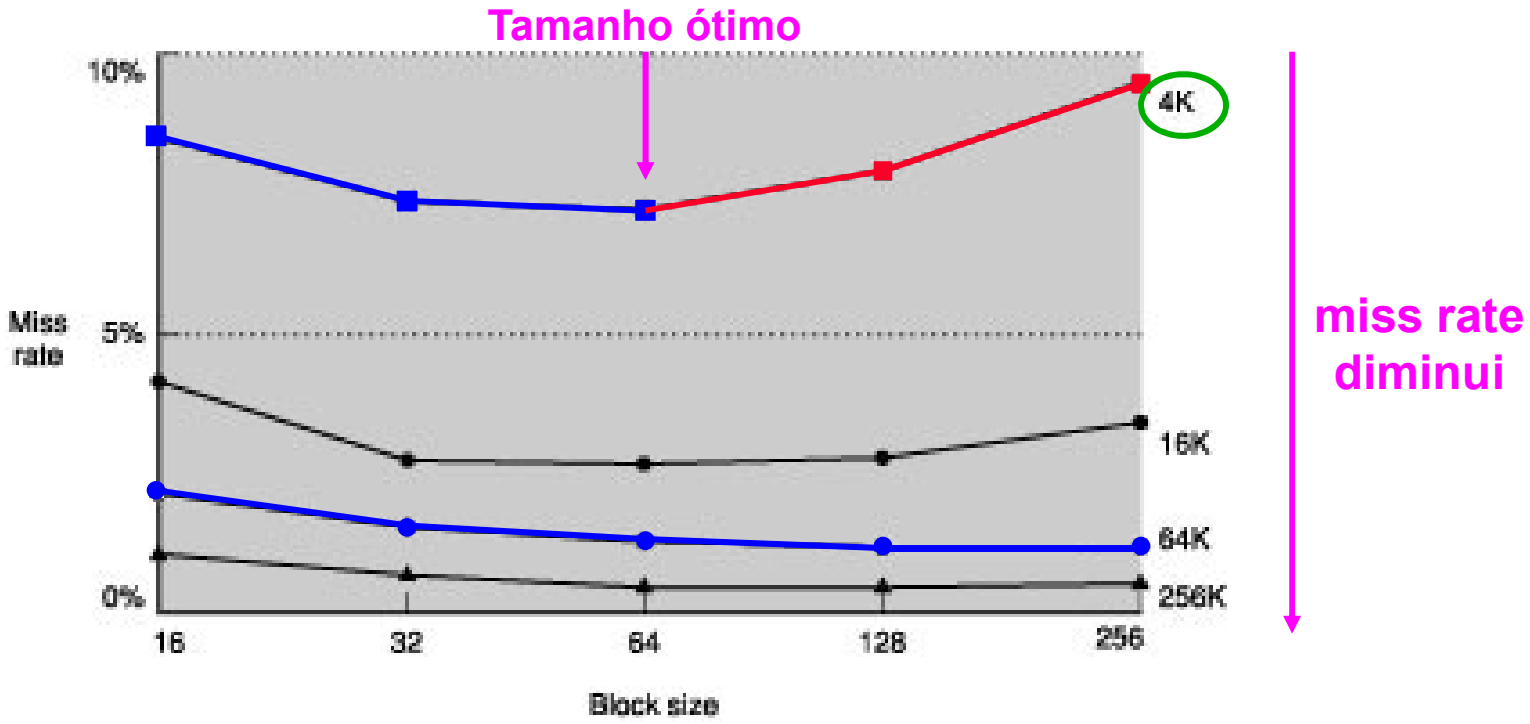
 # bloco mod 64 = 75 mod 64 = 11

1200_{dec} = **010010110000**

...

1215_{dec} = **010010111111**

Impacto do tamanho do bloco



↑ bloco
captura local. espacial ↑
miss rate ↓

MAS,
capacidade fixa

↑bloco → num. blocos ↓
competição p/ bloco ↑
captura local. temporal ↓
miss rate ↑

Manipulando Faltas na Cache

- **Unidade de controle monitora entrada "hit"**
  - **Detecta falta ou acerto**
- **Acerto**
  - **Continua execução normal**
- **Falta**
  - **Pausa da CPU ("stall on miss")**
    - » **Unidade de controle da CPU "congela" registradores**
    - » **Controlador dedicado copia item requisitado p/ cache**

Diferenças no “stall”

- **Pausa devido a falta na cache**
  - **Toda a CPU sofre pausa**
    - » **Conteúdo dos registradores é “congelado”**
- **Pausa devido a hazard**
  - **Nem toda a CPU sofre pausa**
    - » **Algumas instruções continuam execução**
    - » **Para resolver o hazard**

Manipulando faltas na cache

- Enviar endereço para MP
- Comandar leitura da MP
- Esperar que acesso se complete
  - Requer múltiplos ciclos
- Atualizar a cache:
  - Bloco buscado na MP → campo de dados
  - MSBs do endereço → tag
  - Bit de validade é ativado
- Re-iniciar acesso à cache
  - Agora, item será encontrado!

Falta no acesso a instrução

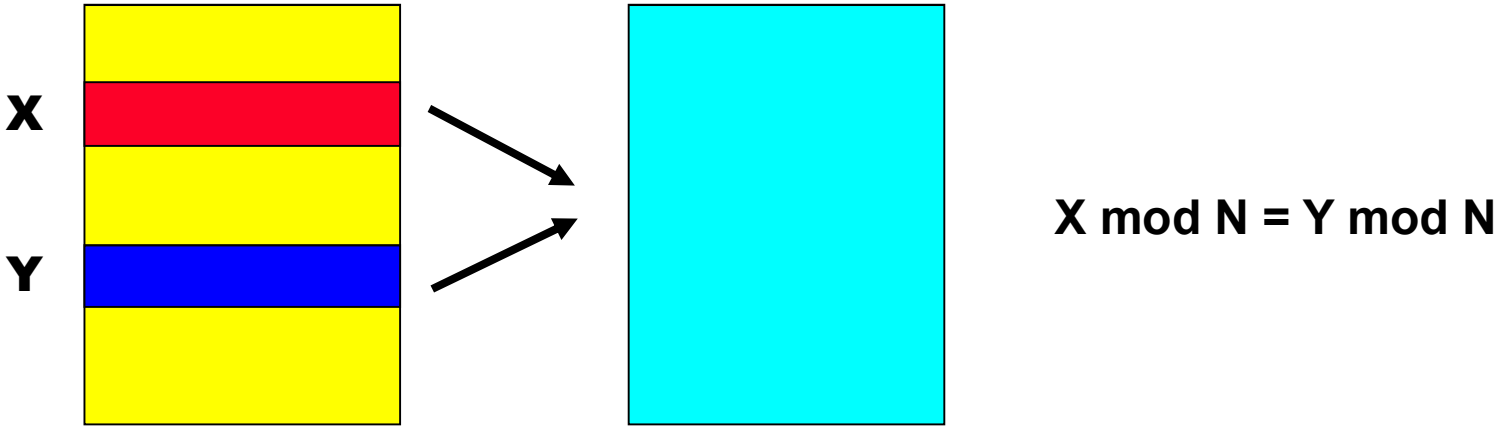
- Enviar endereço para MP: atual PC-4
- Comandar leitura da MP
- Esperar que acesso se complete
  - Requer múltiplos ciclos
- Atualizar a cache:
  - Bloco buscado na MP → campo de dados
  - MSBs do endereço → tag
  - Bit de validade é ativado
- Re-iniciar acesso à cache: IF
  - Agora, item será encontrado! (instrução)

Falta no acesso a dado

- Enviar endereço para MP: endereço efetivo
- Comandar leitura da MP
- Esperar que acesso se complete
  - Requer múltiplos ciclos
- Atualizar a cache:
  - Bloco buscado na MP → campo de dados
  - MSBs do endereço → tag
  - Bit de validade é ativado
- Re-iniciar acesso à cache: MEM
  - Agora, item será encontrado! (dado)

Faltas na escrita

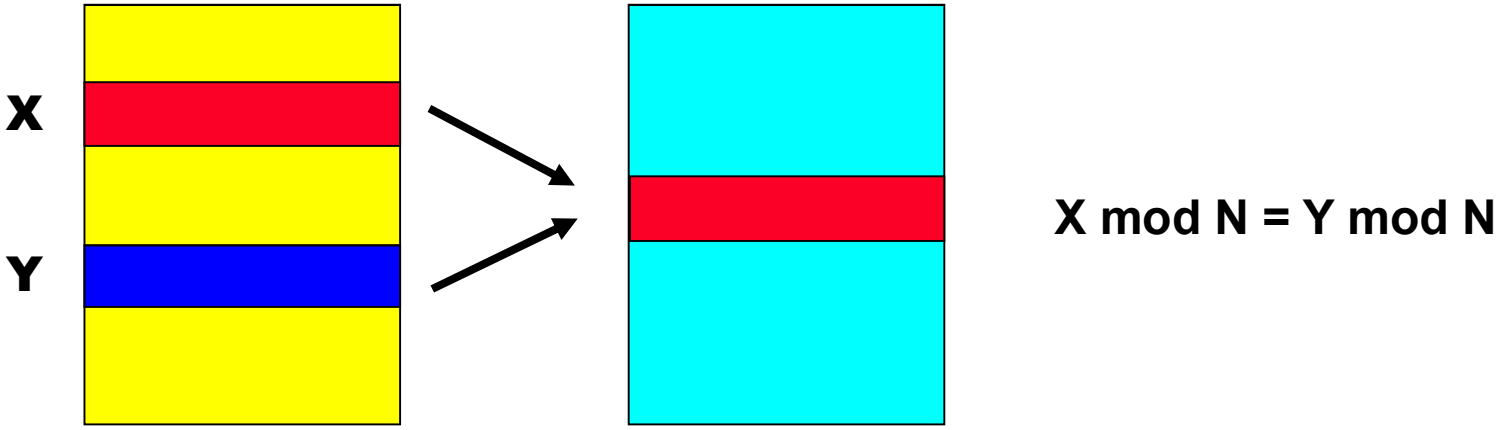
- **Hipótese: m=0 e mapeamento direto**



- **Se houver um acesso de escrita em X?**
  - **O valor de X é escrito no bloco**

Faltas na escrita

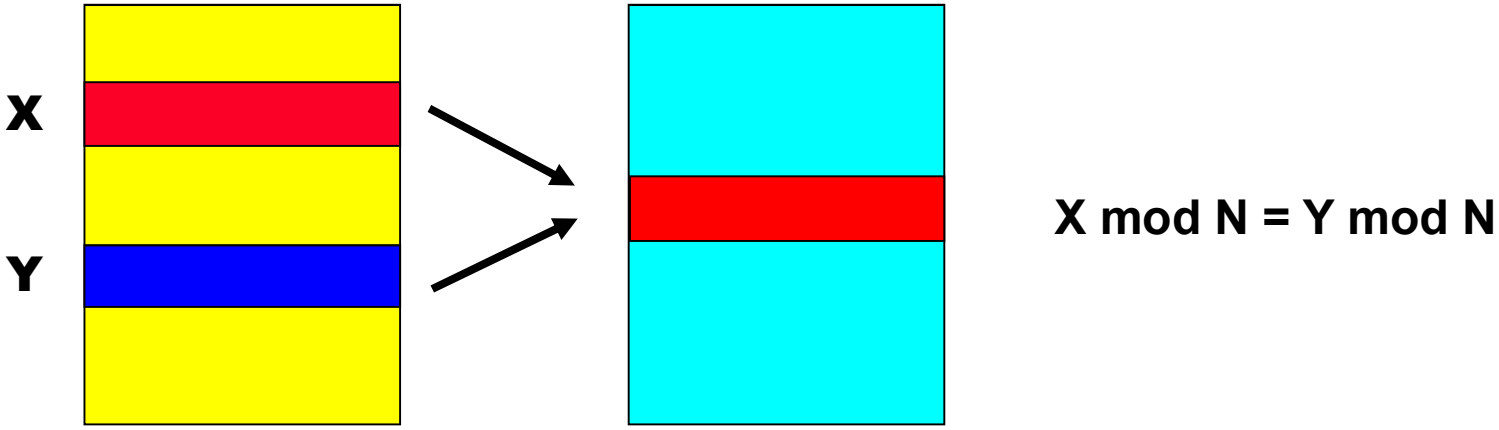
- **Hipótese: m=0 e mapeamento direto**



- **Se houver um acesso de escrita em X?**
  - **O valor de X é escrito no bloco**

Faltas na escrita

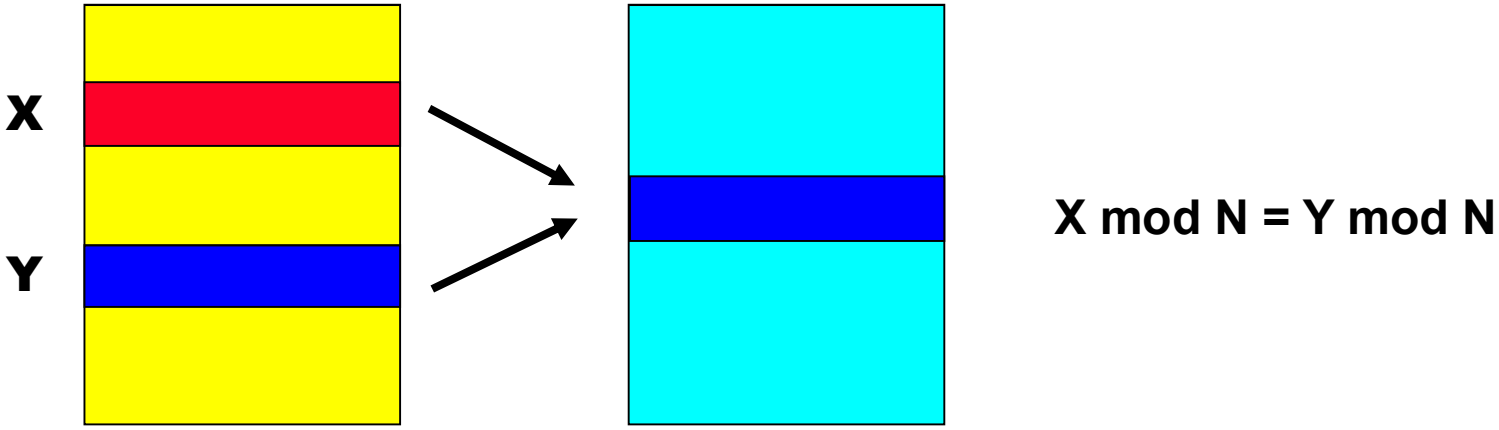
- **Hipótese: m=0 e mapeamento direto**



- **Se houver um acesso de escrita em X?**
  - **O valor de X é escrito no bloco**
- **E se, depois, houver um acesso de escrita em Y?**
  - **O valor de Y é escrito no bloco**

Faltas na escrita

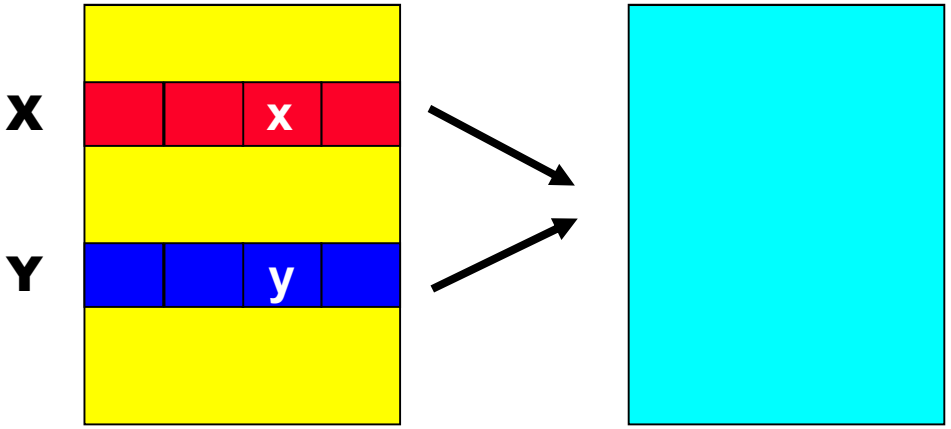
- **Hipótese: m=0 e mapeamento direto**



- **Conclusão: Escrita independente de qual endereço**
  - **Não é preciso comparar tag na escrita**
  - **Basta escrever o item (e seu tag) na cache**
- **Mas este é caso particular:**
  - **E se houvesse múltiplas palavras no bloco?**

Faltas na escrita

- **Hipótese: m>0 e mapeamento direto**

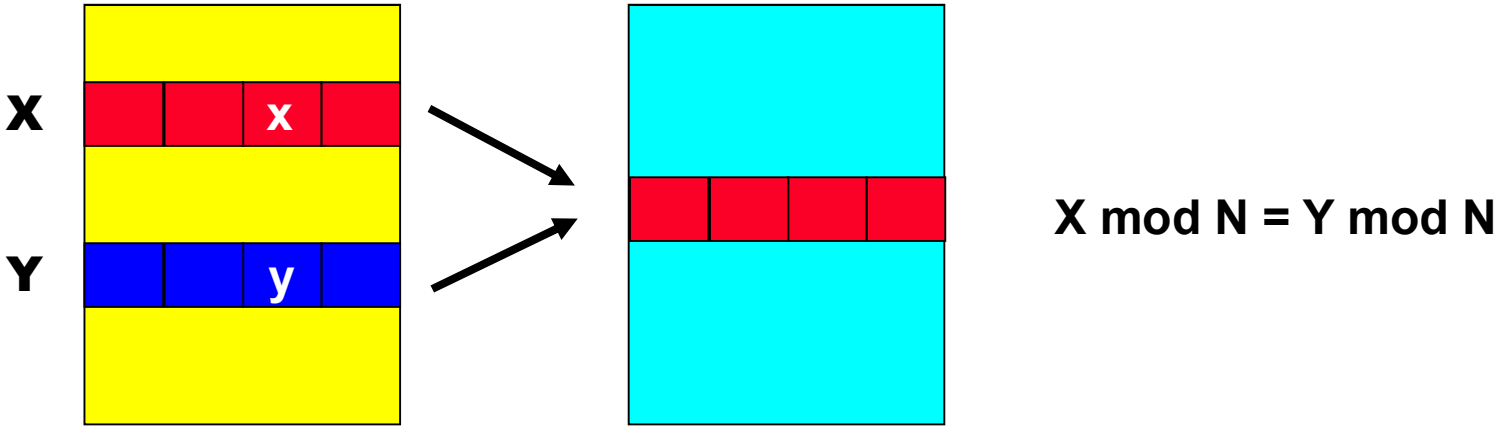


X mod N = Y mod N

- **Se houver um acesso de escrita em x?**

Faltas na escrita

- **Hipótese: m>0 e mapeamento direto**

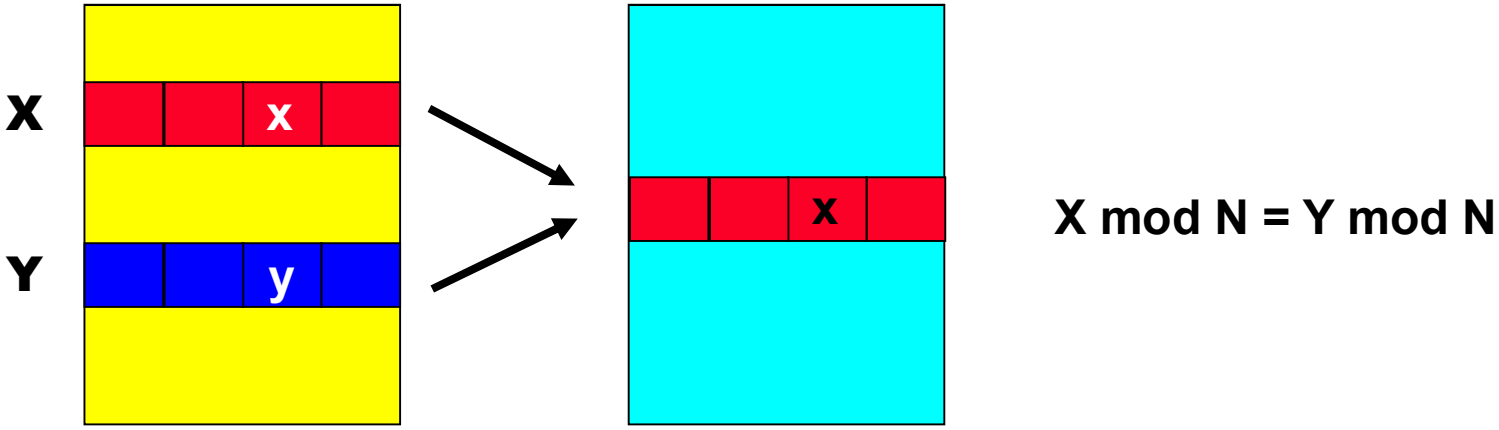


X mod N = Y mod N

- **Se houver um acesso de escrita em x?**
  - **O bloco X é lido**

Faltas na escrita

- **Hipótese: m>0 e mapeamento direto**

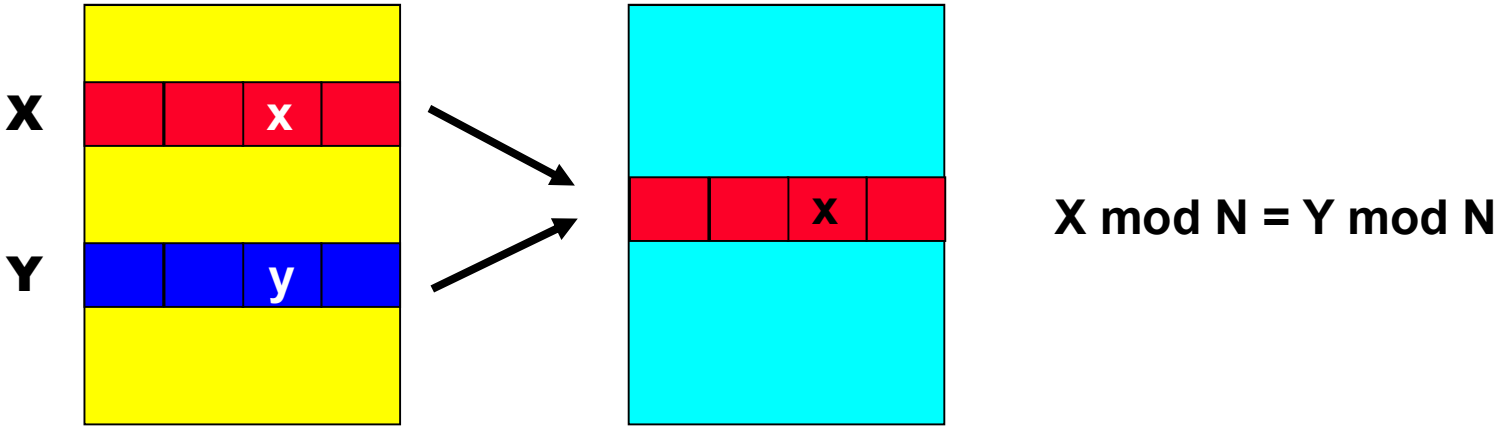


X mod N = Y mod N

- **Se houver um acesso de escrita em x?**
  - **O bloco X é lido e o valor de x é escrito**

Faltas na escrita

- **Hipótese: m>0 e mapeamento direto**

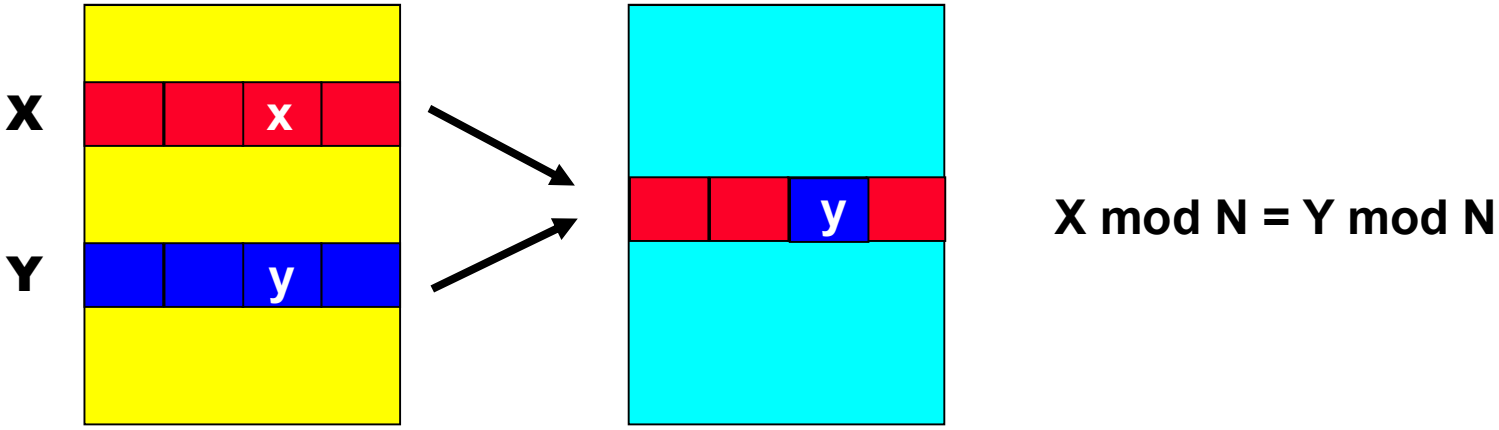


X mod N = Y mod N

- **Se houver um acesso de escrita em x?**
  - **O bloco X é lido e o valor de x é escrito**
- **E se, depois, houver um acesso de escrita em y?**
  - **O valor de y pode simplesmente ser escrito?**

Faltas na escrita

- **Hipótese: m>0 e mapeamento direto**

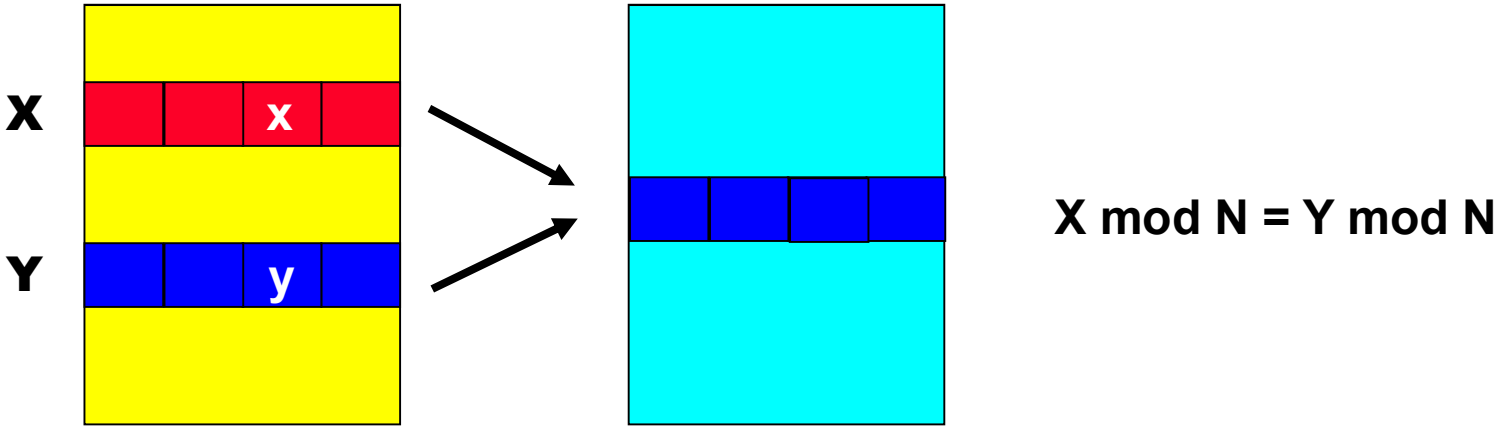


X mod N = Y mod N

- **Não!**

Faltas na escrita

- **Hipótese: m>0 e mapeamento direto**

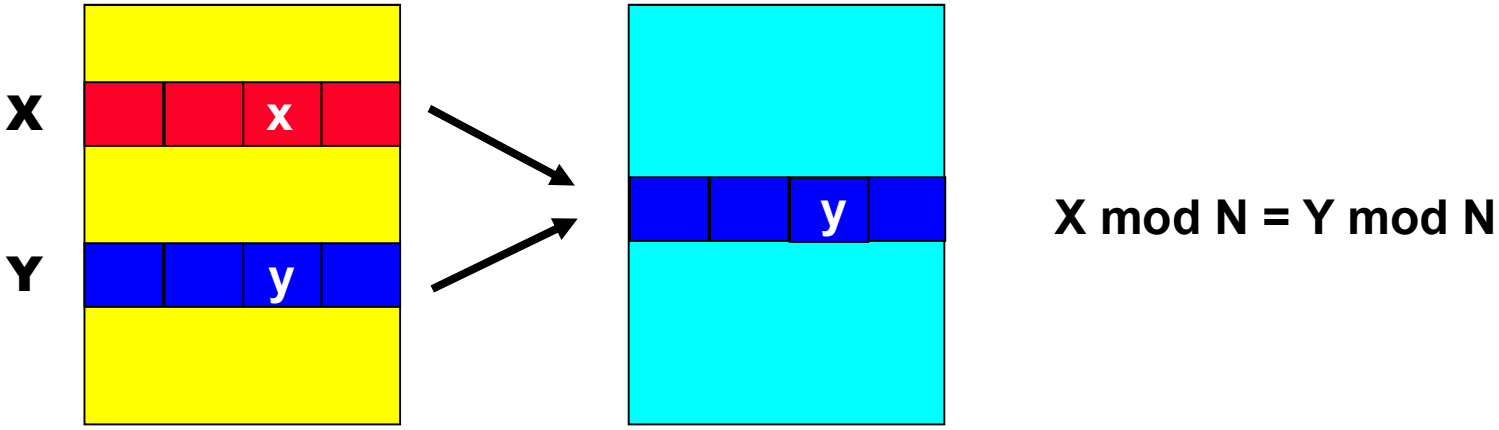


X mod N = Y mod N

- **Não!**
- **Bloco Y inteiro precisa ser lido**

Faltas na escrita

- **Hipótese: m>0 e mapeamento direto**



X mod N = Y mod N

- **Não!**
- **Bloco Y inteiro precisa ser lido**
  - **Antes de se escrever o valor de y na cache**
  - **Falta na escrita em cache requerer leitura na MP!**
    - » **"Write allocate"**

Manipulando escritas na cache

- **Discussão:**
  - **Se store escrevesse só na cache**
    - » **Sem alterar MP**
  - **Diferentes valores na MP e na cache**
    - » **Inconsistência**
- **Mecanismo para manter consistência**
  - **"Write-through"**
    - » **Store sempre escreve em ambas: MP e cache**

Manipulando escritas na cache

- **Ineficiência do "write-through"**
  - **MP é muito mais lenta que cache**
    - » **Instruções store seriam muito mais lentas que outras**
- **Exemplo:**
  - **Tempo de acesso (MP) = 100 ciclos**
  - **SPECInt2000: 10% são stores**
  - **Ausência de hazards**
  - **CPI = 1 + 0,1 x 100 = 11**
    - » **Máquina ficaria ~10 vezes mais lenta**

Manipulando escritas na cache

- **Solução paliativa: "Write buffer"**
  - **Armazena escritas pendentes (valor, endereço)**
- **Dinâmica da escrita**
  - **Store escreve na cache e no "write buffer"**
  - **CPU continua execução (próximas instruções)**
  - **Quando uma escrita na MP se completa**
    - » **Sua entrada no write buffer é liberada**
- **Limitação**
  - **CPU sofre pausa quando "write buffer" cheio**

Manipulando escritas na cache

- **Alternativa: "write-back"**
  - **Store escreve só no bloco da cache**
  - **Bloco atualizado na MP só quando estiver na iminência de ser substituído**
- **Vantagens**
  - **Menor impacto da "bandwidth" no desempenho**
    - » **Menor número de escritas em MP (ou cache L_{i+1})**
    - » **Várias escritas em MP (ou L_{i+1}) "filtradas" pela L_i**
  - **Menor consumo de energia**
    - » **Menor chaveamento em barramentos mais capacitivos**

Manipulando escritas na cache

- **Desvantagens**
  - **Implementação mais complexa**
    - » Verificação de “dirty bit” no controlador de cache
    - » Mais estados e transições
  - **Requer gerenciamento de valores inconsistentes**
    - » Valor em MP (ou L_{i+1}) diferente p/ E/S ou p/ outro core
    - » Ações para garantir consumo do valor correto em MP (ou L_{i+1})
  - **Pode resultar em desempenho inferior**
    - » WB: paradas no pipeline quando bloco atualizado na MP durante execução
    - » WT+Wbuffer: escrita na MP em paralelo com execução
- **Uso típico**
  - **“Write-through” combinado com “write-buffer”**
  - **“Write-back” puro**

Exemplo real

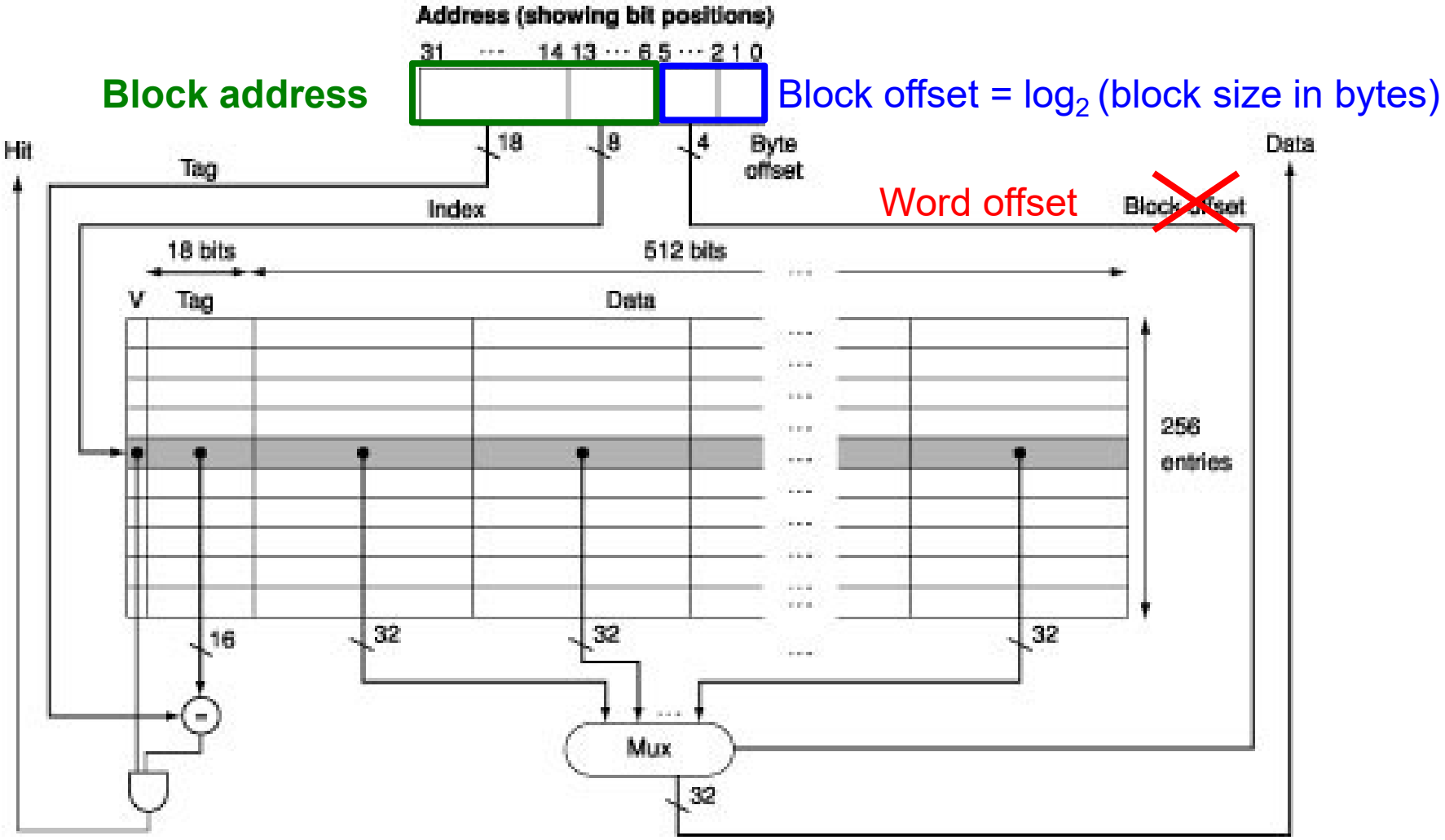
- **Instrinsity FastMATH**
  - **Processador embarcado = MIPS + cache**
- **Pipeline**
  - **12 estágios**
  - **Pode requisitar 1 instrução e 1 dado simultâneos**
    - » **Caches separadas: I-cache e D-cache**
    - » **Sinais independentes para read e write em cada cache**
- **Cache**
  - **16KB (cada)**
  - **Bloco: 16 palavras**

#Blocos = 4K palavras/16 palavras =256

n = 8 (índice)

m = 4 (word offset)

FastMATH: estrutura



FastMATH: comportamento

- **Leitura**
  - **Enviar endereço à cache apropriada**
    - » **PC (I-cache) ou ALU (D-cache)**
  - **Acerto: palavra requisitada disponível**
    - » **Selecionada pelo MUX**
  - **Falta: endereço enviado à MP**
    - » **Bloco copiado na cache e, só então, lido**
- **Escrita**
  - **Mecanismo escolhido pelo sistema operacional**
    - » **“Write through” + “write buffer” de 1 entrada**
    - » **“Write back”**

FastMATH: desempenho da cache

- **SPEC2000 benchmarks**

Instruction miss rate	Data miss rate	Effective combined miss rate
0.4%	11.4%	3.2%

(Supõe que loads+stores = 34% I)

- **Discussão:**
  - **Quem explora mais localidade, I ou D-cache?**
  - **Qual tem menor taxa de faltas …**
    - » **Cache unificada ou caches separadas?**
  - **Unificada (m = 3,18%); separadas (m = 3,2%)**
    - » **Separadas: maior "bandwidth" e sem hazards estruturais**
    - » **Unificada: redução insignificante nas paradas devidas a faltas; aumento substancial de paradas devidas a hazards**