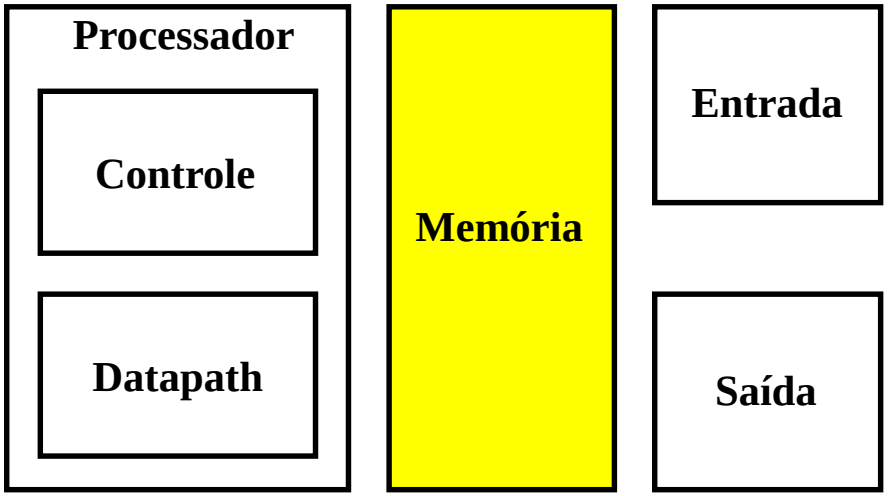
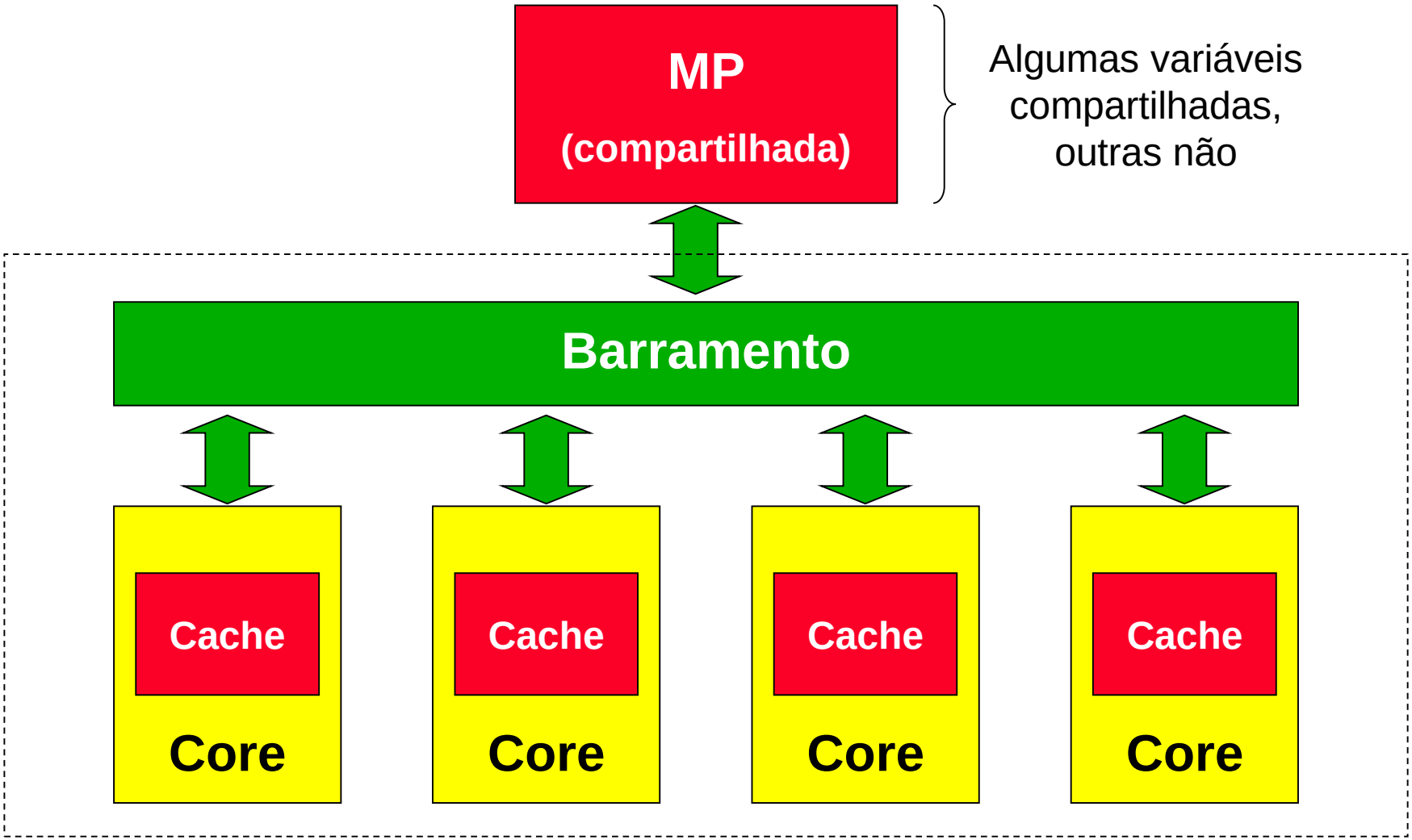
Paralelismo e hierarquias de memória: coerência de cache



CMP: “on-chip multiprocessing”



O problema da coerência de cache

- **Hipótese:**
  - **Uma única posição de memória (X)**
  - **Lida e escrita por dois processadores (A e B)**
  - **“Write through”**

Time step	Event	Cache contents for CPU A	Cache contents for CPU B	Memory contents for location X

O problema da coerência de cache

- **Hipótese:**
  - **Uma única posição de memória (X)**
  - **Lida e escrita por dois processadores (A e B)**
  - **“Write through”**

Time step	Event	Cache contents for CPU A	Cache contents for CPU B	Memory contents for location X
0				0

O problema da coerência de cache

- **Hipótese:**
  - **Uma única posição de memória (X)**
  - **Lida e escrita por dois processadores (A e B)**
  - **"Write through"**

Time step	Event	Cache contents for CPU A	Cache contents for CPU B	Memory contents for location X
0				0
1	CPU A reads X	0		0

O problema da coerência de cache

- **Hipótese:**
  - **Uma única posição de memória (X)**
  - **Lida e escrita por dois processadores (A e B)**
  - **"Write through"**

Time step	Event	Cache contents for CPU A	Cache contents for CPU B	Memory contents for location X
0				0
1	CPU A reads X	0		0
2	CPU B reads X	0	0	0

O problema da coerência de cache

- **Hipóteses:**
  - **Uma única posição de memória (X)**
  - **Lida e escrita por dois processadores (A e B)**
  - **"Write through"**

Time step	Event	Cache contents for CPU A	Cache contents for CPU B	Memory contents for location X
0				0
1	CPU A reads X	0		0
2	CPU B reads X	0	0	0
3	CPU A stores 1 into X	1	0	1

Coerência e consistência

- **Um sistema de memória é coerente quando a leitura de um item retorna o valor mais recentemente escrito naquele item.**
- **Definição é vaga e simplista**
- **Na verdade, contém dois diferentes aspectos do comportamento do sistema de memória**
- **Coerência:**
  - **Quais valores podem ser retornados por uma leitura**
- **Consistência:**
  - **Quando um valor escrito será retornado por uma leitura**

Coerência e consistência

- **Um sistema de memória é coerente quando a leitura de um item retorna o valor mais recentemente escrito naquele item?**
- **Definição é simplista**
- **Na verdade, contém dois diferentes aspectos do comportamento do sistema de memória**
- **Coerência: (conceito explicado nesta aula)**
  - **Quais valores podem ser retornados por uma leitura**
- **Consistência: (conceito fora do escopo de INE5411)**
  - **Quando um valor escrito será retornado por uma leitura**

Coerência: requisito 1

- **O sistema de memória é coerente se preserva a ordem do programa localmente, ou seja:**
- **Processador P escreve na posição X**
- **Nenhum outro processador faz escrita intermediária em X**
- **Processador P lê a posição X e o valor retornado é o próprio valor escrito por P**

Coerência: requisito 2

- **O sistema de memória é coerente se provê globalmente uma visão coerente da memória, ou seja:**
- **Processador P1 escreve na posição X**
- **Nenhum outro processador faz escrita intermediária em X**
- **O próximo acesso de leitura a X está suficientemente afastado de sua escrita**
- **Processador P2 lê a posição X e o valor retornado é o valor escrito por P1**

Coerência: requisito 3

- **O sistema de memória é coerente se provê uma mesma ordem de escrita para todos os processadores, ou seja:**
- **Processador P1 escreve na posição X**
- **Processador P2 escreve na posição X**
- **Leituras de X feitas por P1 ou P2 enxergam os valores escritos na mesma ordem**

Técnicas para garantir coerência

- **Chave para implementação: detectar o**
  - **Status de compartilhamento de um bloco**
    - » **Mais de uma cópia do bloco em cache distintas**

Técnicas para garantir coerência

- **Chave para implementação: detectar o**
  - **Status de compartilhamento de um bloco**
    - » **Mais de uma cópia do bloco em cache distintas**
- **Protocolos de coerência de cache**
  - **Distribuído: "Snooping"**
    - » **Cada controlador de cache monitora o status**
      - **Verifica se cache local tem cópia do bloco referenciado**
    - » **Uso: pequeno número de "cores" (2, 4)**

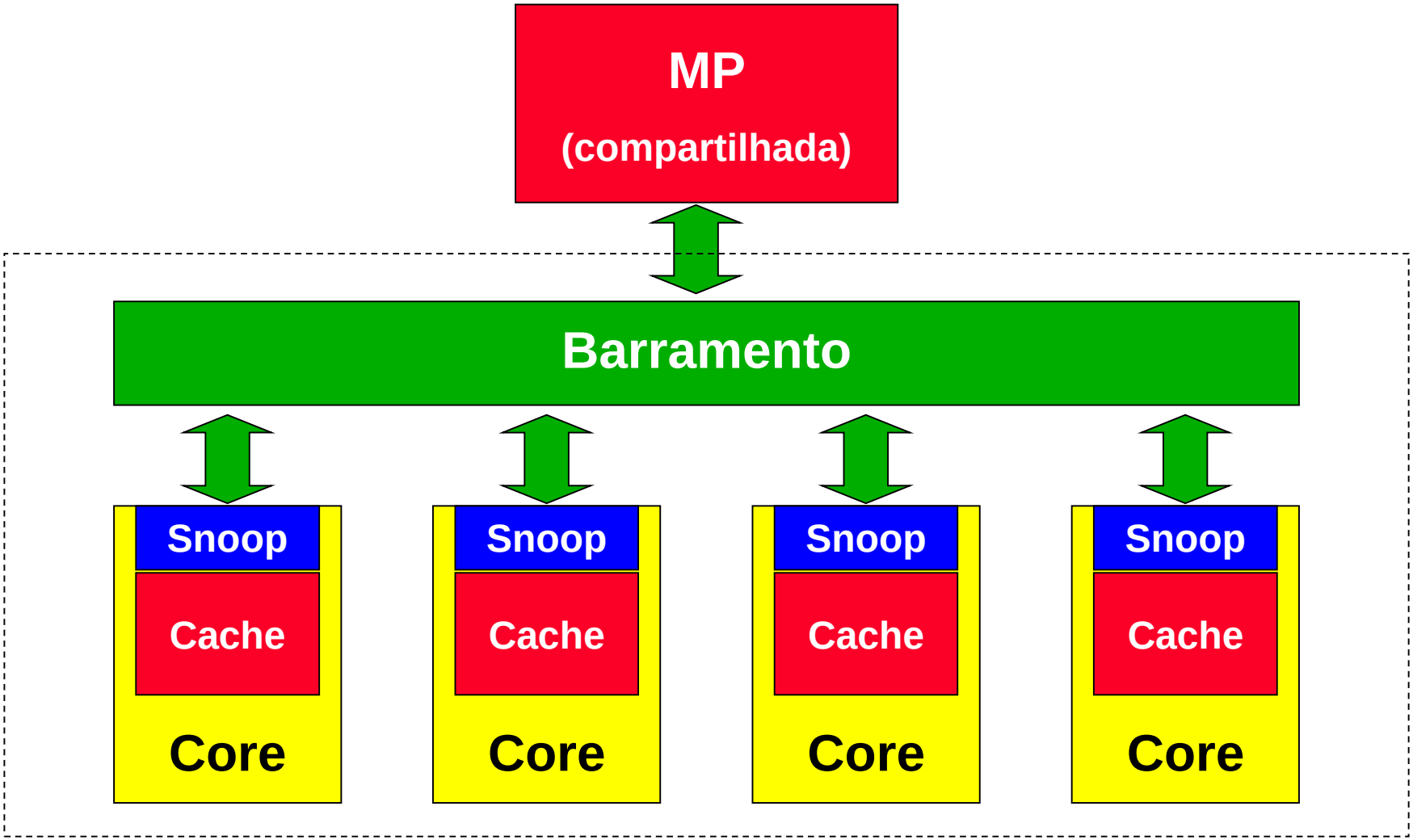
Técnicas para garantir coerência

- **Chave para implementação: detectar o**
  - **Status de compartilhamento de um bloco**
    - » **Mais de uma cópia do bloco em cache distintas**
- **Protocolos de coerência de cache**
  - **Distribuído: "Snooping"**
    - » **Cada controlador de cache monitora o status**
      - **Verifica se cache local tem cópia do bloco referenciado**
    - » **Uso: pequeno número de "cores" (2, 4)**
  - **Centralizado: Diretório**
    - » **O status é mantido em um único lugar**
      - **Para cada bloco, um vetor de bits indica qual core tem uma cópia**
    - » **Uso: grande número de "cores" (8, 16, ...)**

“Snooping”

- **Requisito 1: meio compartilhado que faz “broadcasting” de faltas na cache**
  - **Exemplo: barramento compartilhado**
- **Requisito 2: controladores de cache são estendidos para “espionar” o barramento**

CMP: “on-chip multiprocessing”



“Invalidação”

- **Ideia-chave:**
  - **Ao escrever um item da cache, um processador deve invalidar outras cópias daquele item em outras caches**
- **Consequências:**
  - **Protocolo do tipo “write-invalidate”**
  - **Um processador tem exclusividade de acesso a um item antes de nele escrever**

Exemplo da ação do “snooping”

- **Hipóteses:**
  - **Posição X lida e escrita por processadores A e B**
  - **“Write back” (quando bloco torna-se compartilhado)**

Processor activity	Bus activity	Contents of CPU A’s cache	Contents of CPU B’s cache	Contents of memory location X
				0

Inicialmente, nenhuma das caches contém o valor de X

Exemplo da ação do “snooping”

- **Hipóteses:**
  - **Posição X lida e escrita por processadores A e B**
  - **“Write back” (quando bloco torna-se compartilhado)**

Processor activity	Bus activity	Contents of CPU A’s cache	Contents of CPU B’s cache	Contents of memory location X
				0
CPU A reads X	Cache miss for X	0		0

A lê X

Exemplo da ação do "snooping"

- **Hipóteses:**
  - **Posição X lida e escrita por processadores A e B**
  - **"Write back" (quando bloco torna-se compartilhado)**

Processor activity	Bus activity	Contents of CPU A's cache	Contents of CPU B's cache	Contents of memory location X
				0
CPU A reads X	Cache miss for X	0		0
CPU B reads X	Cache miss for X	0	0	0

B lê X

Exemplo da ação do "snooping"

- **Hipóteses:**
  - **Posição X lida e escrita por processadores A e B**
  - **"Write back" (quando bloco torna-se compartilhado)**

Processor activity	Bus activity	Contents of CPU A's cache	Contents of CPU B's cache	Contents of memory location X
				0
CPU A reads X	Cache miss for X	0		0
CPU B reads X	Cache miss for X	0	0	0
CPU A writes a 1 to X	Invalidation for X	1		0

A escreve no bloco em sua cache e invalida bloco na cache de B

Exemplo da ação do "snooping"

- **Hipóteses:**
  - **Posição X lida e escrita por processadores A e B**
  - **"Write back" (quando bloco torna-se compartilhado)**

Processor activity	Bus activity	Contents of CPU A's cache	Contents of CPU B's cache	Contents of memory location X
				0
CPU A reads X	Cache miss for X	0		0
CPU B reads X	Cache miss for X	0	0	0
CPU A writes a 1 to X	Invalidation for X	1		0
CPU B reads X	Cache miss for X	1	1	1

Falta na cache de B: **A fornece o novo valor a B e o atualiza na MP**

Memória compartilhada coerente

Comunicação entre *threads*:

- **Leitura e escrita de variáveis compartilhadas**
- **Em um mesmo espaço de endereçamento**

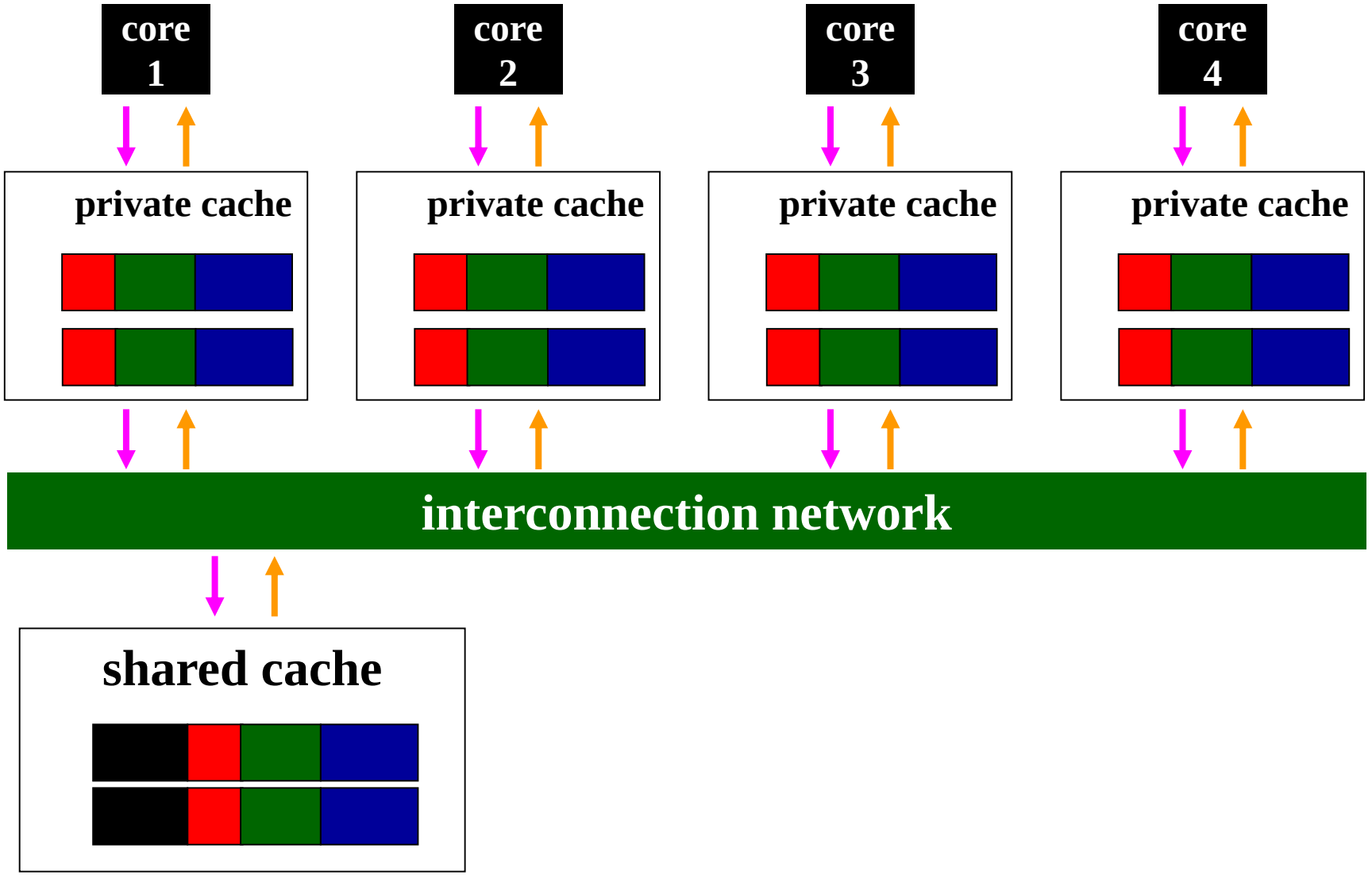
Paradoxo: variável compartilhada em cache privativa

- **Cada core pode ter uma cópia**
- **Modificação de uma cópia induz cópias incoerentes**

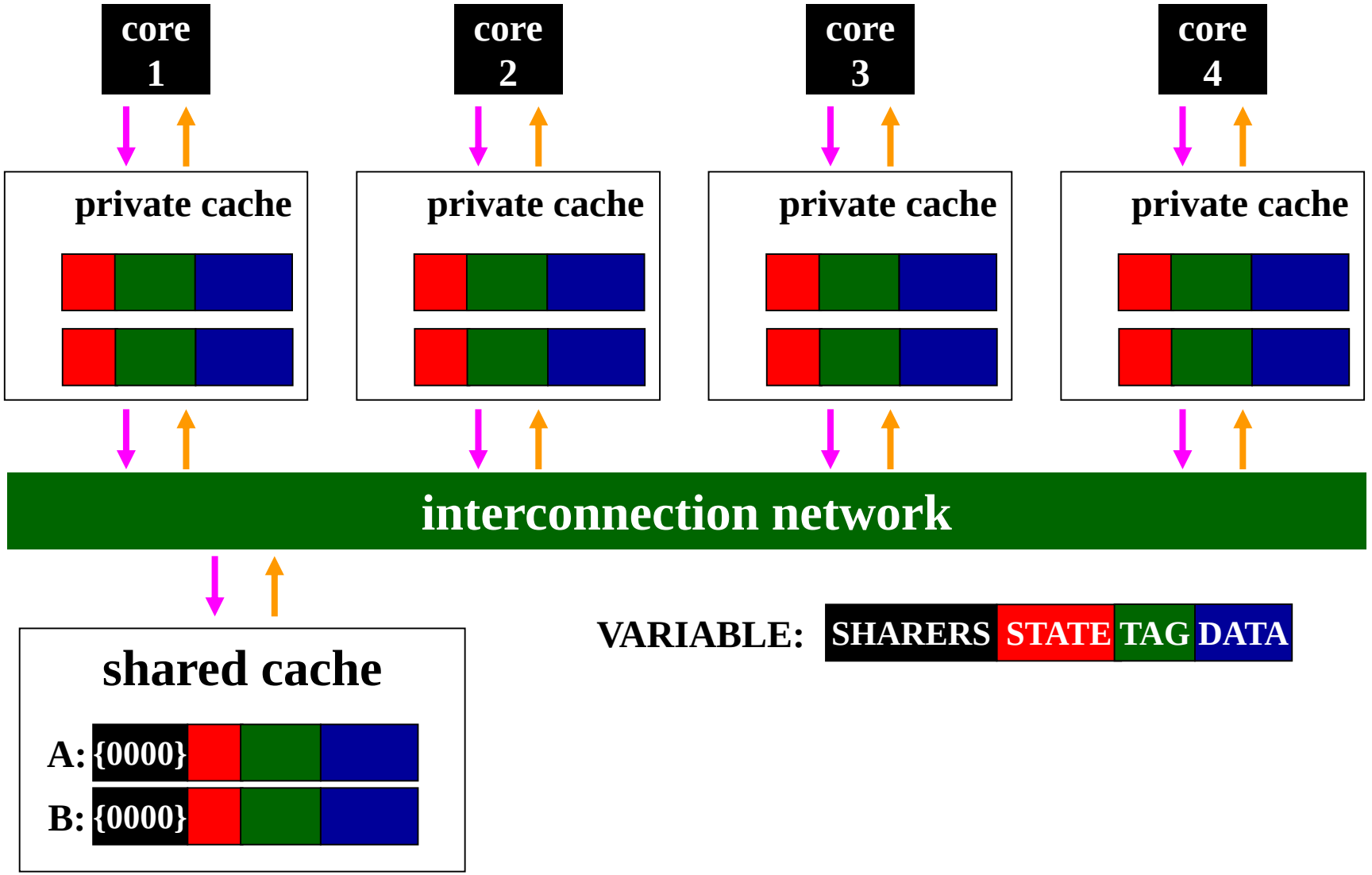
Protocolo de coerência em HW

- **Torna a funcionalidade da cache invisível para o SW**
- **Permite que programador se concentre no que importa**
  - **Encontrar tarefas para serem executadas em paralelo!**
  - **Ao invés de ficar gerenciando a memória**
  - **Sem se preocupar com impacto na sincronização e comunicação**
- **Como funciona um protocolo baseado em diretório?**

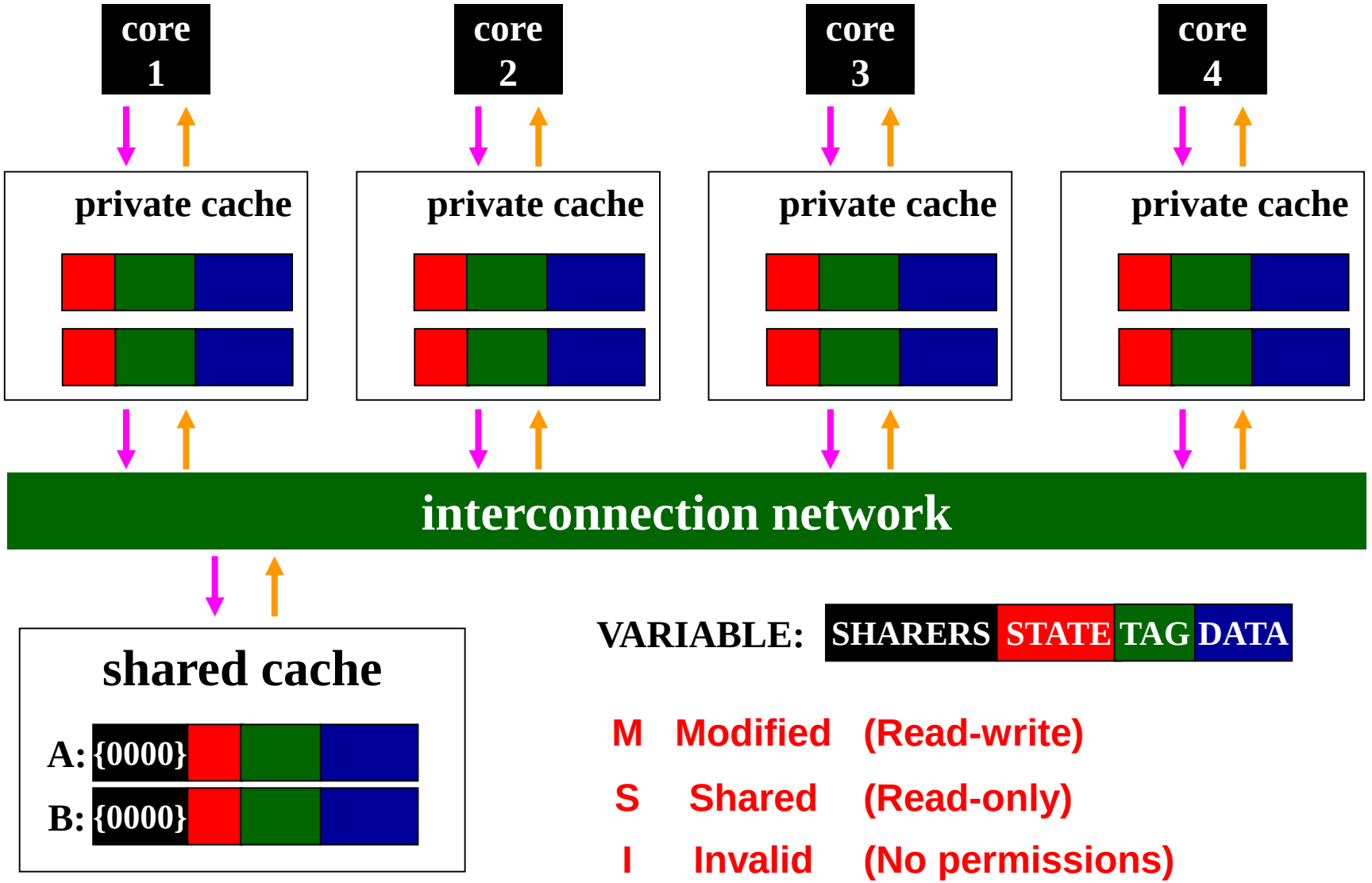
Acesso acelerado via caches



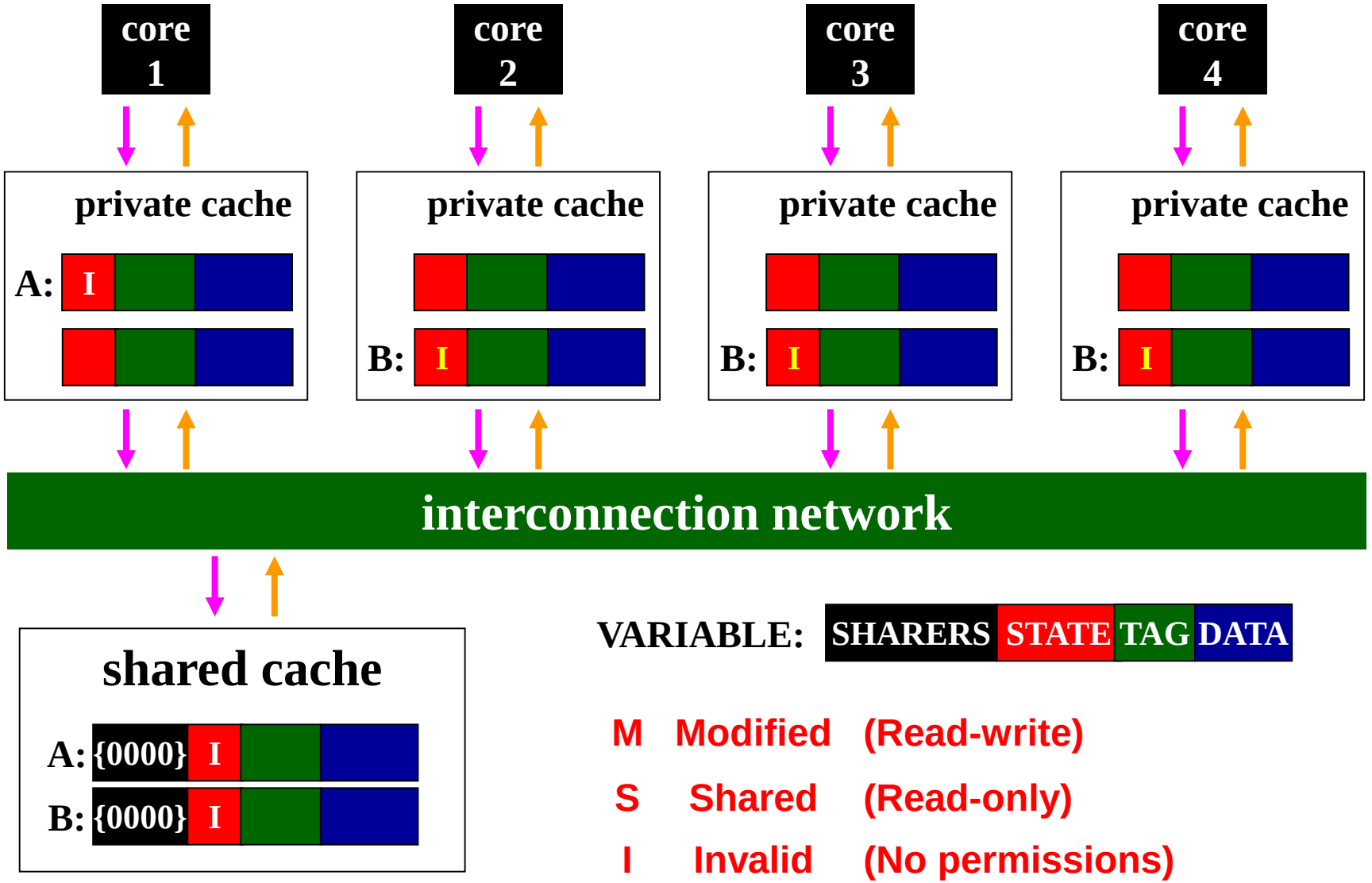
Gerência de múltiplas cópias



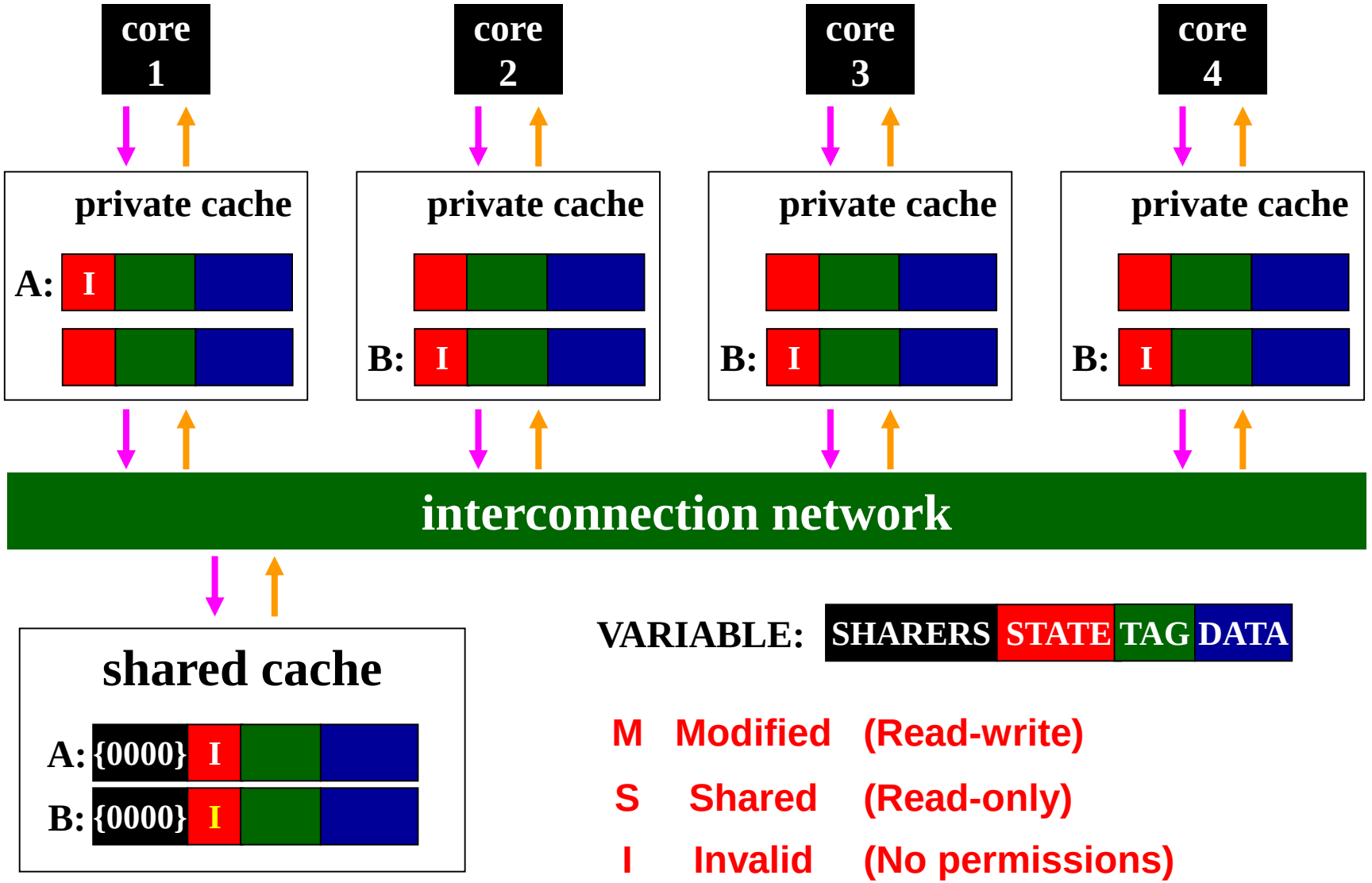
Estado de uma cópia de variável



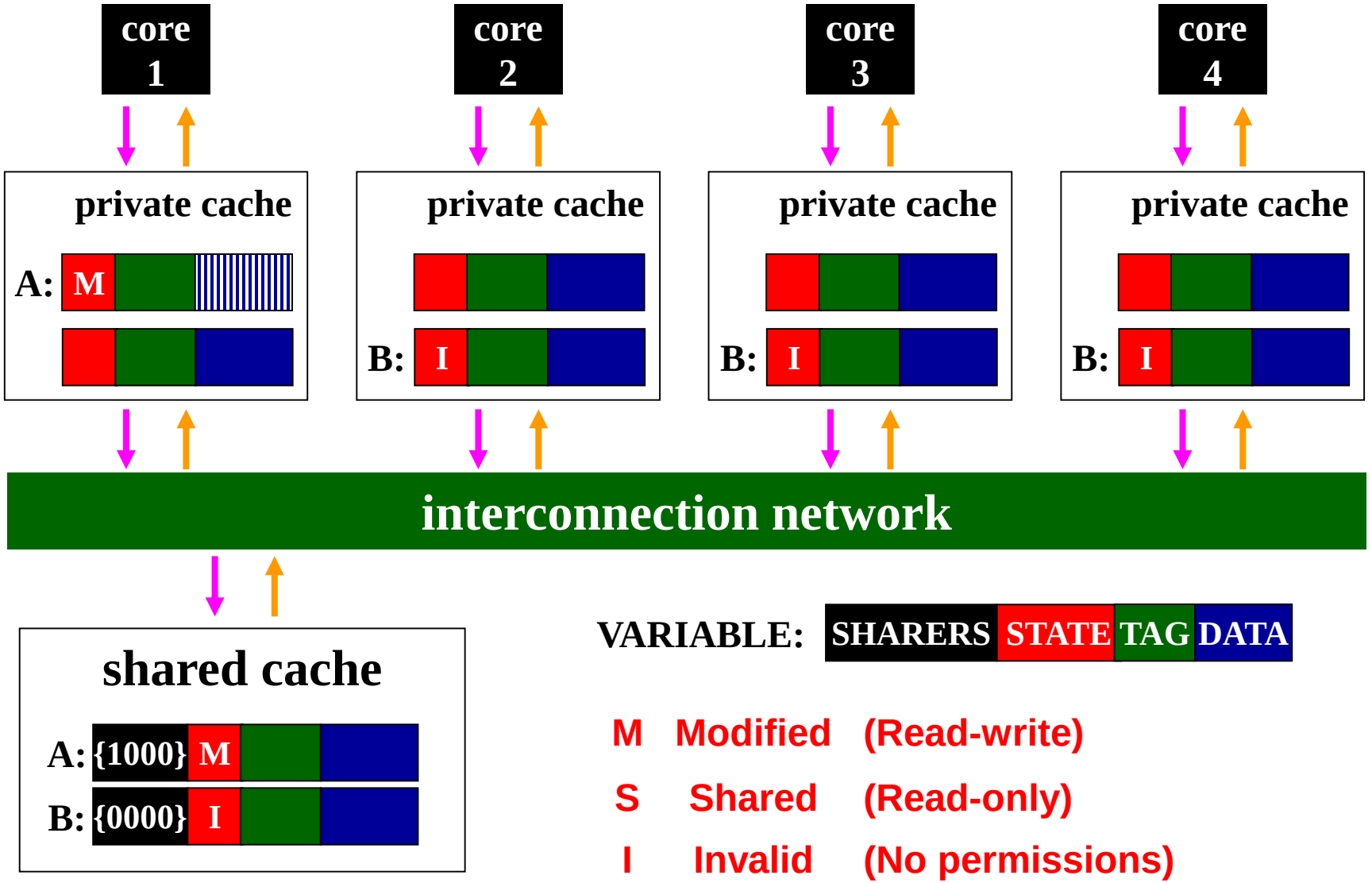
Estado de cópia em *private cache*



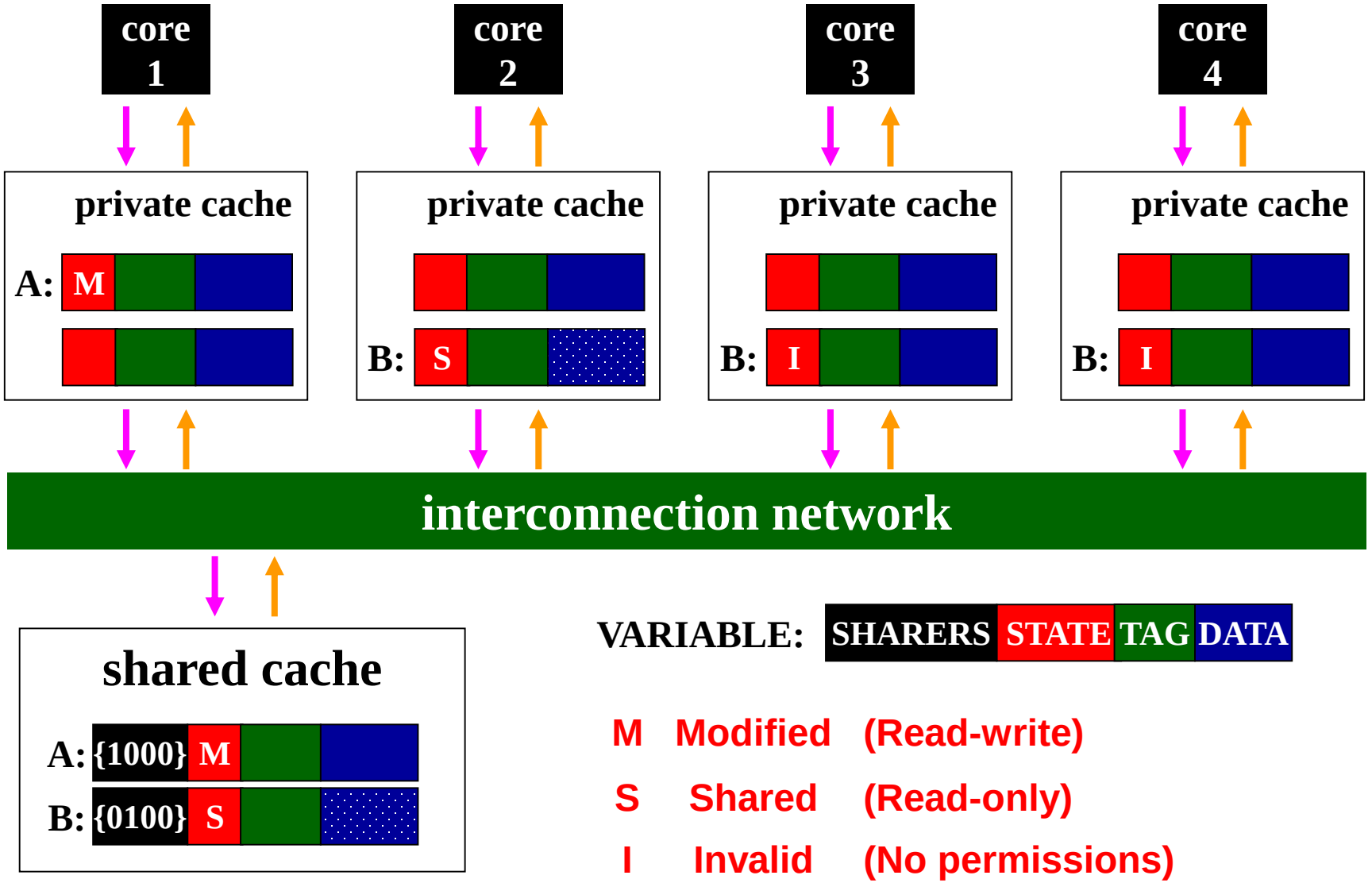
Estado agregado em *shared cache*



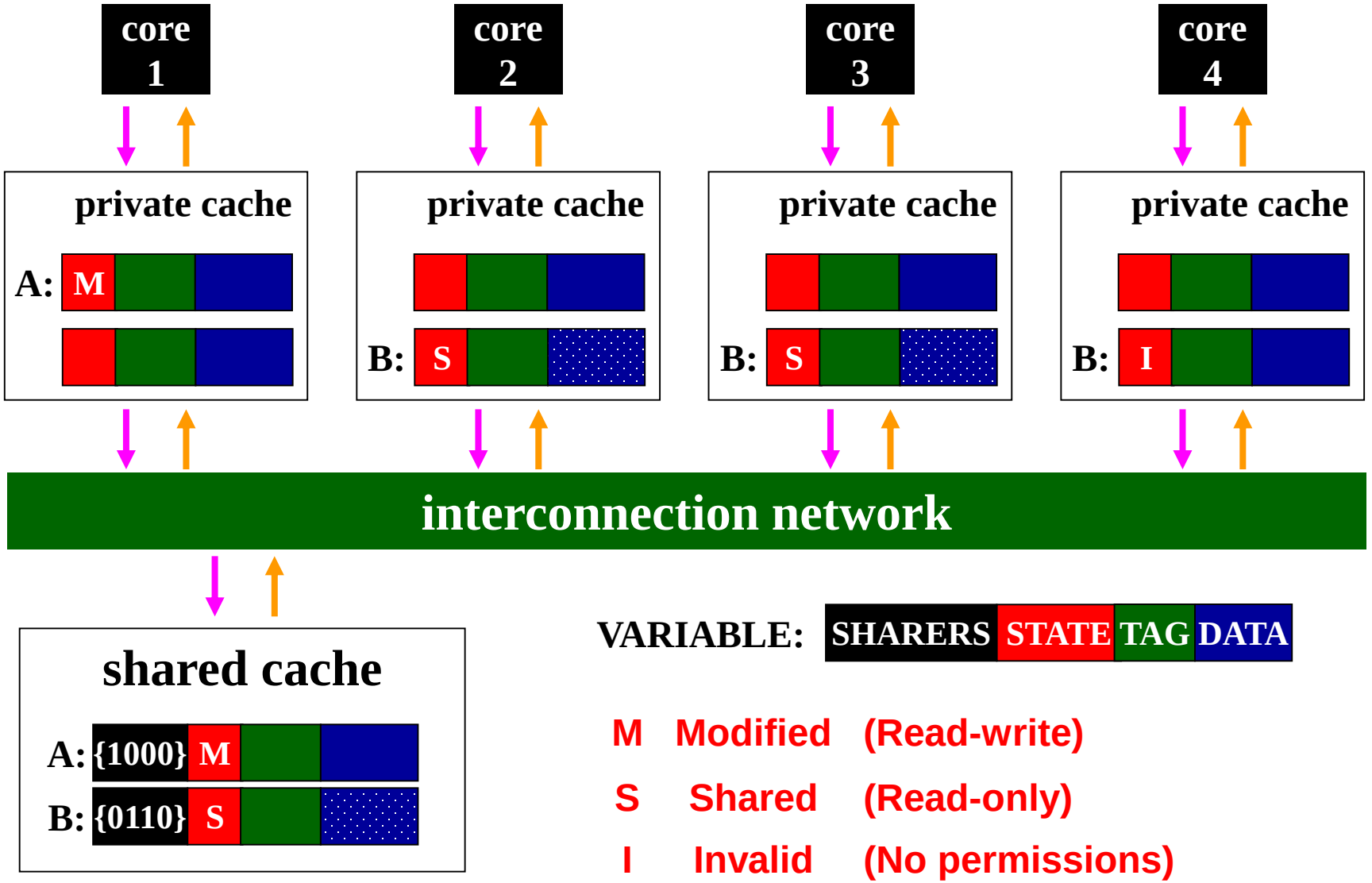
Core 1 escreve na variável A



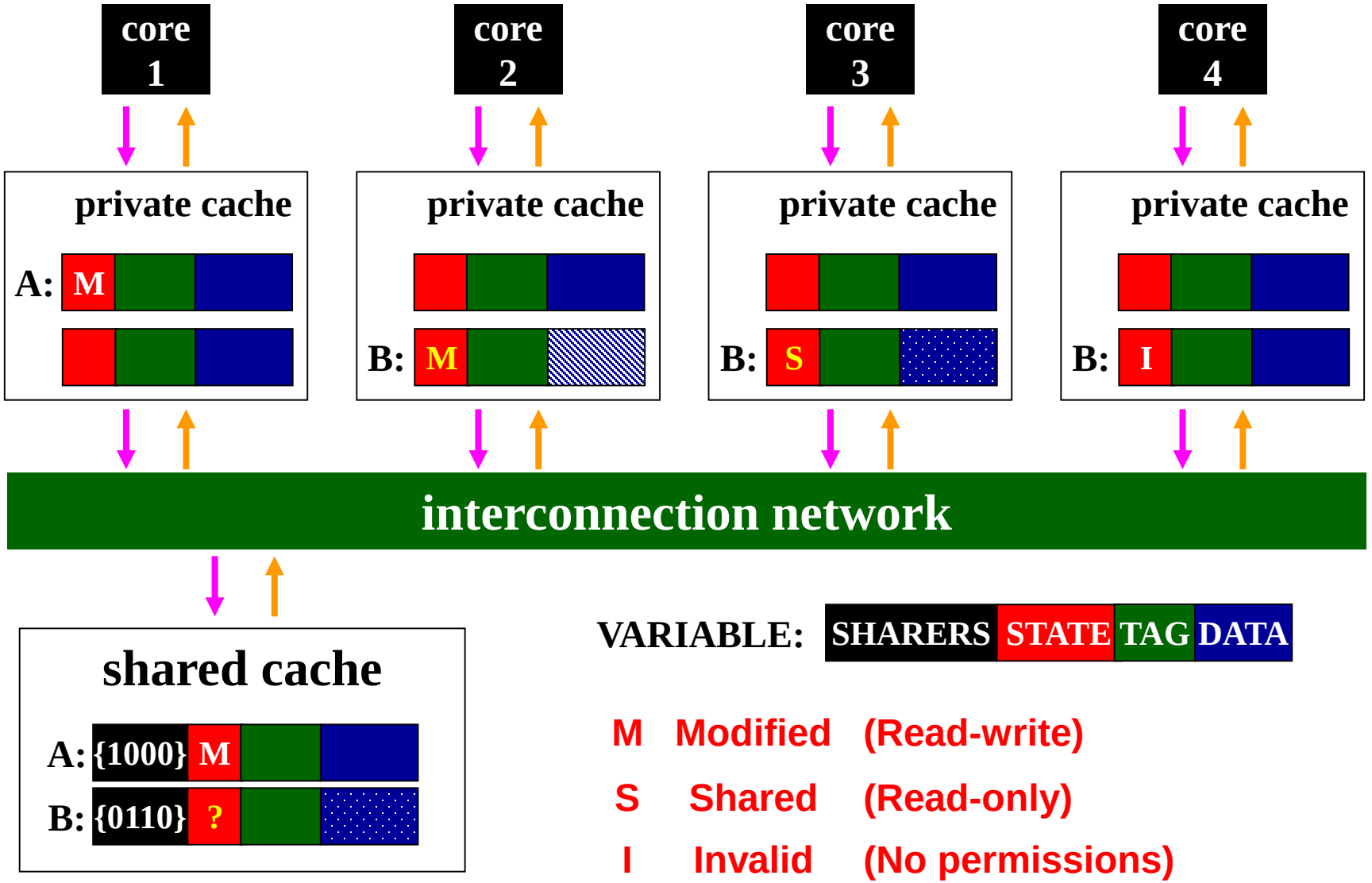
Core 2 lê variável B



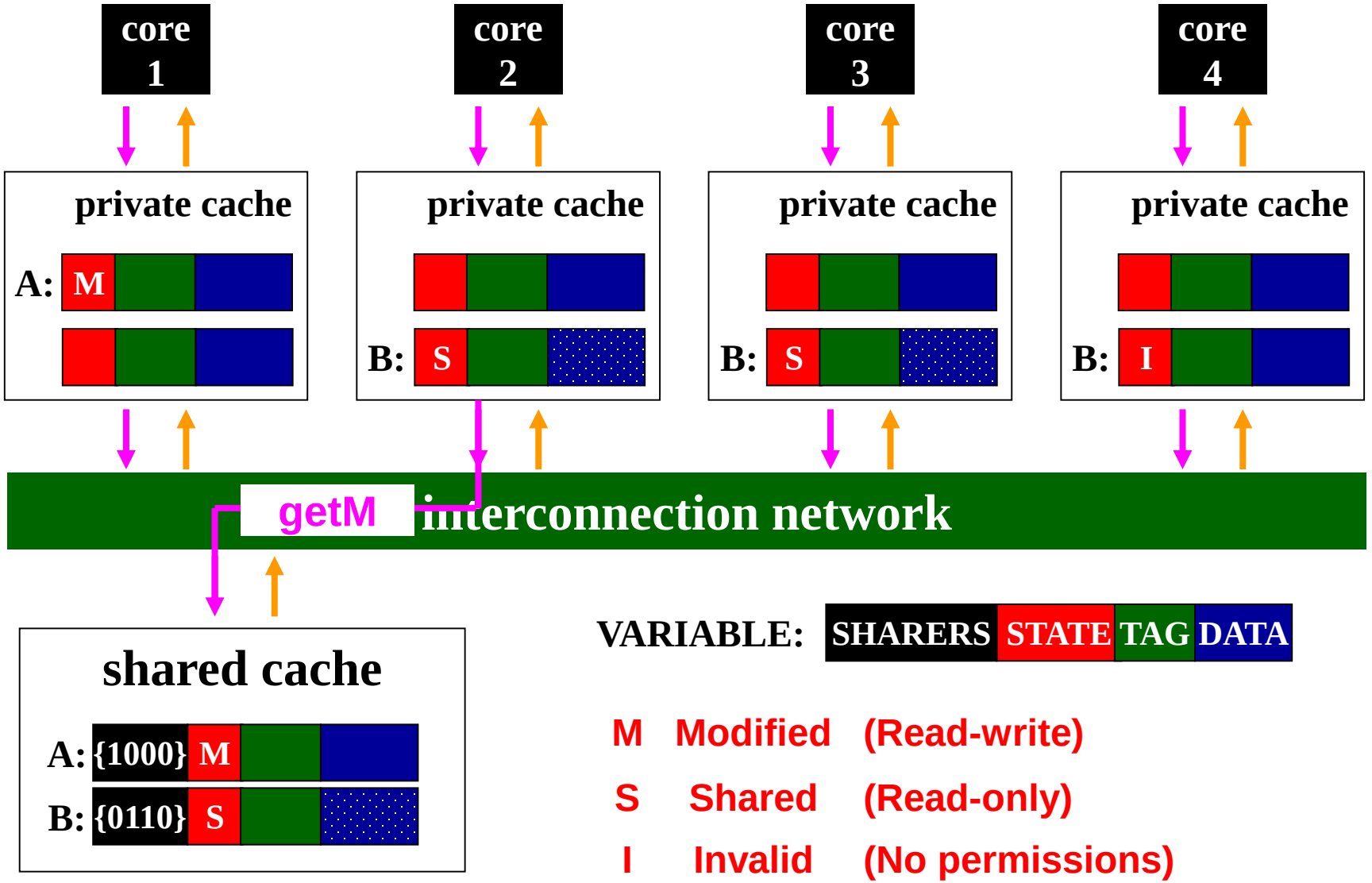
Core 3 lê variável B



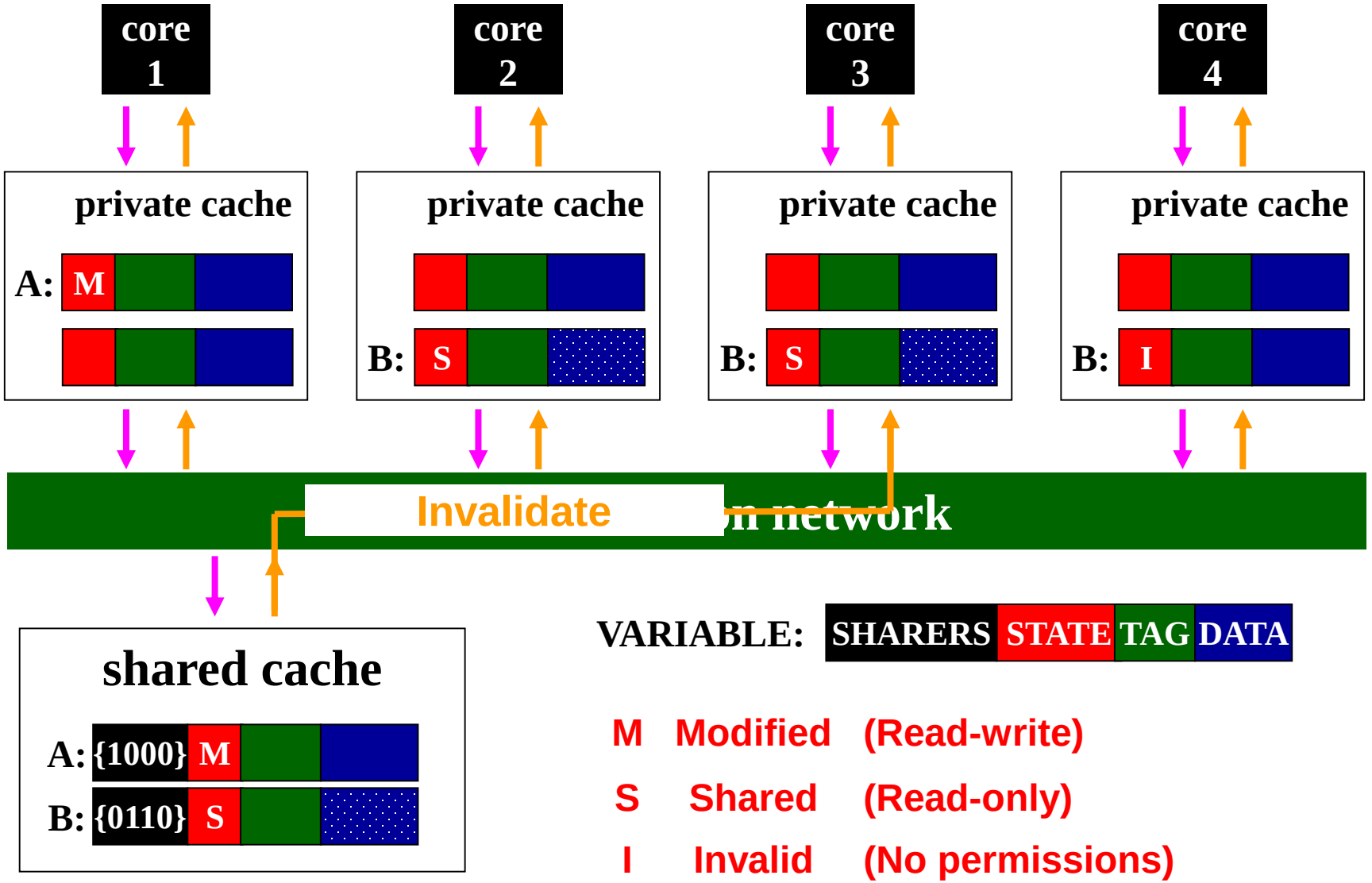
Core 2 escreve em B (incoerência)



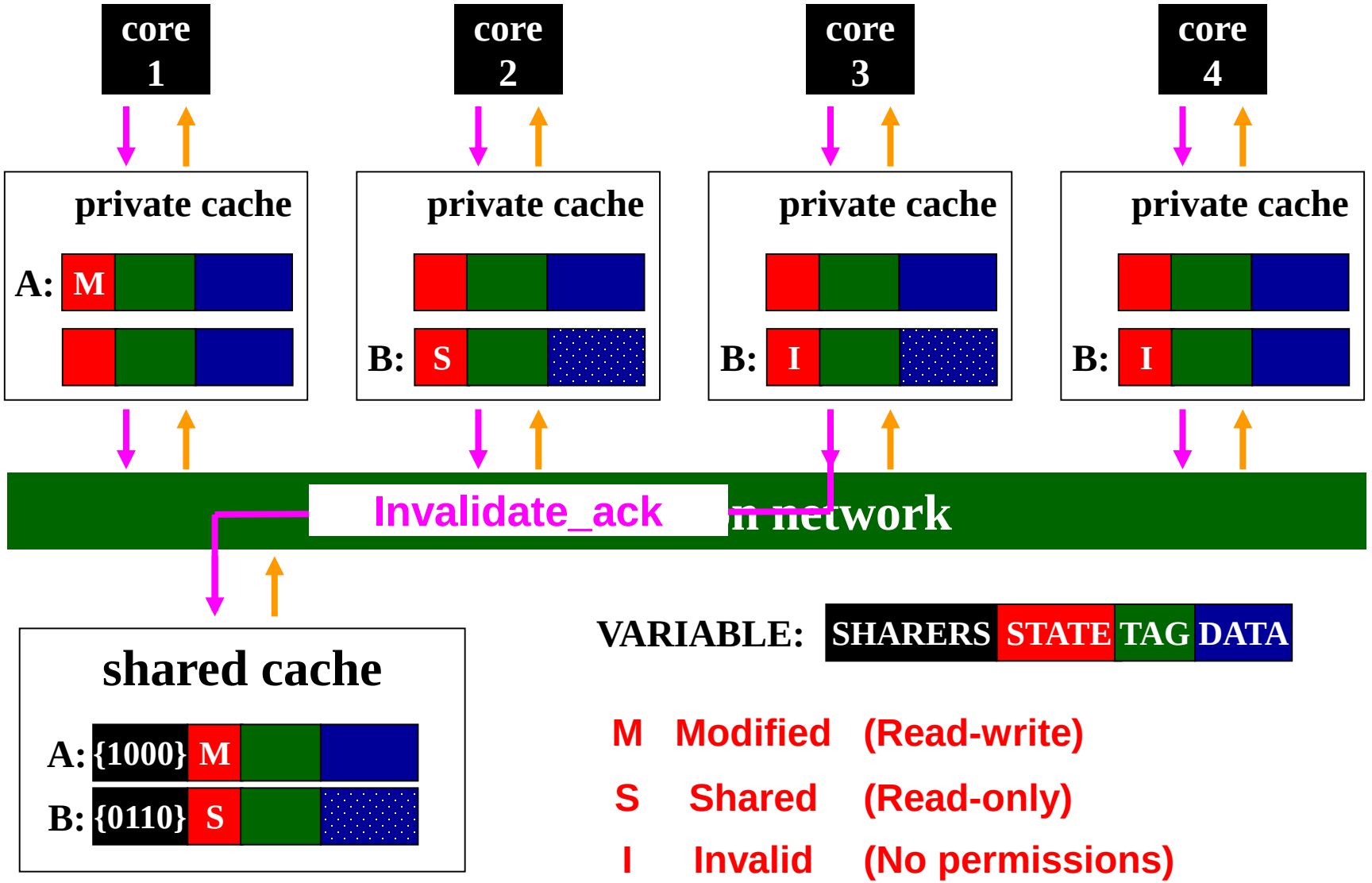
Core 2 pede permissão para escrever



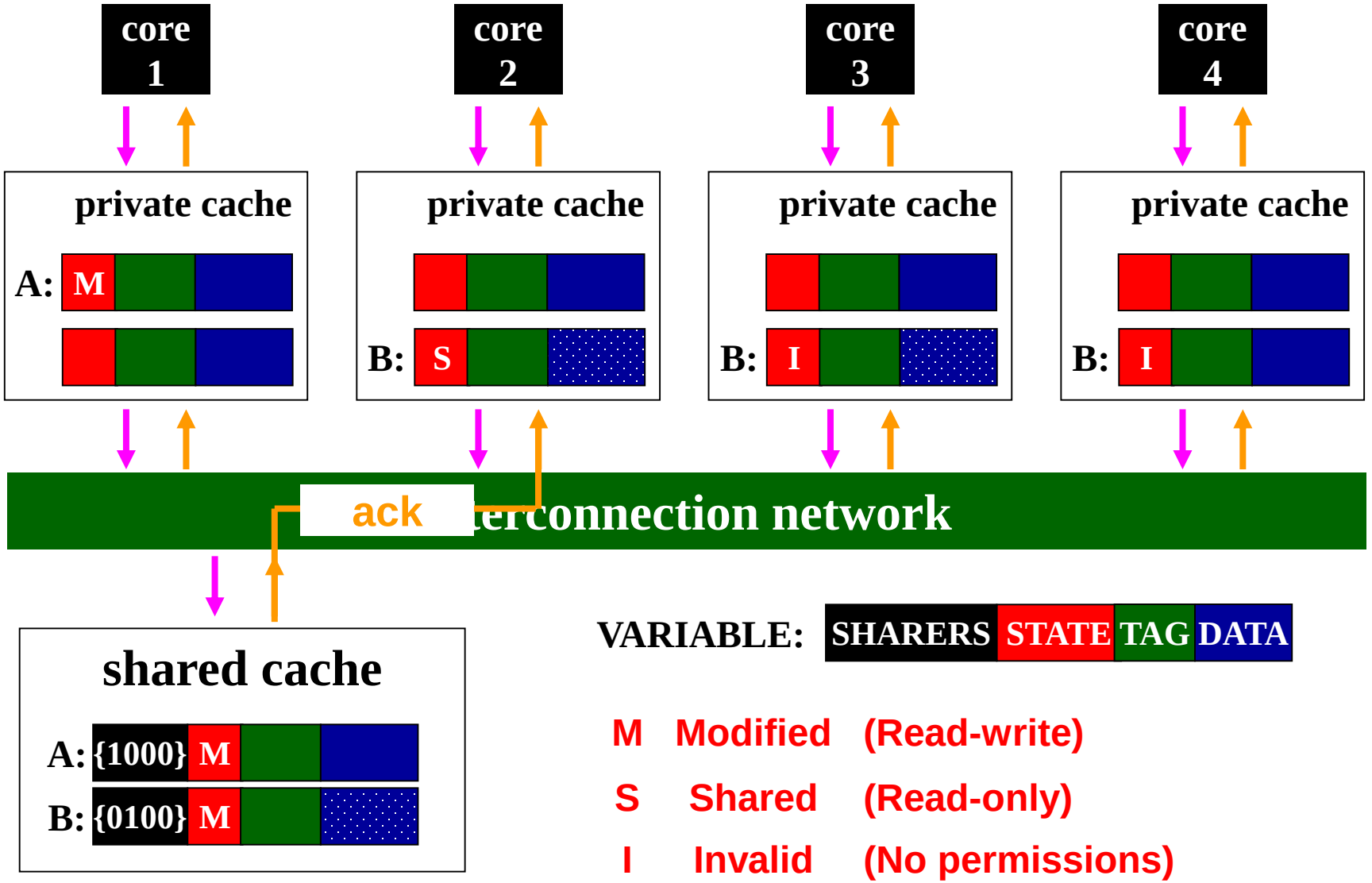
Shared cache envia invalidação



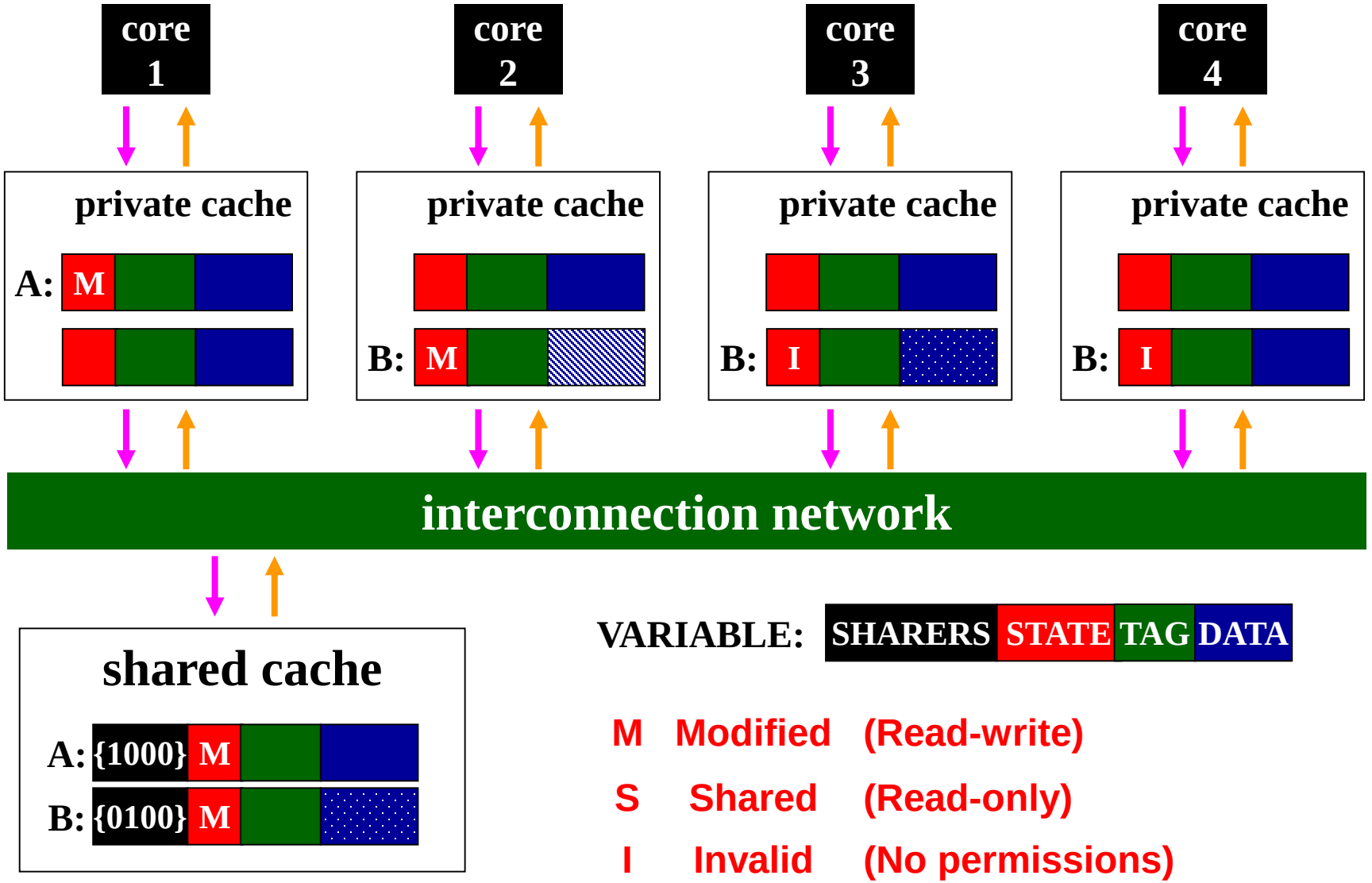
Private cache reconhece invalidação



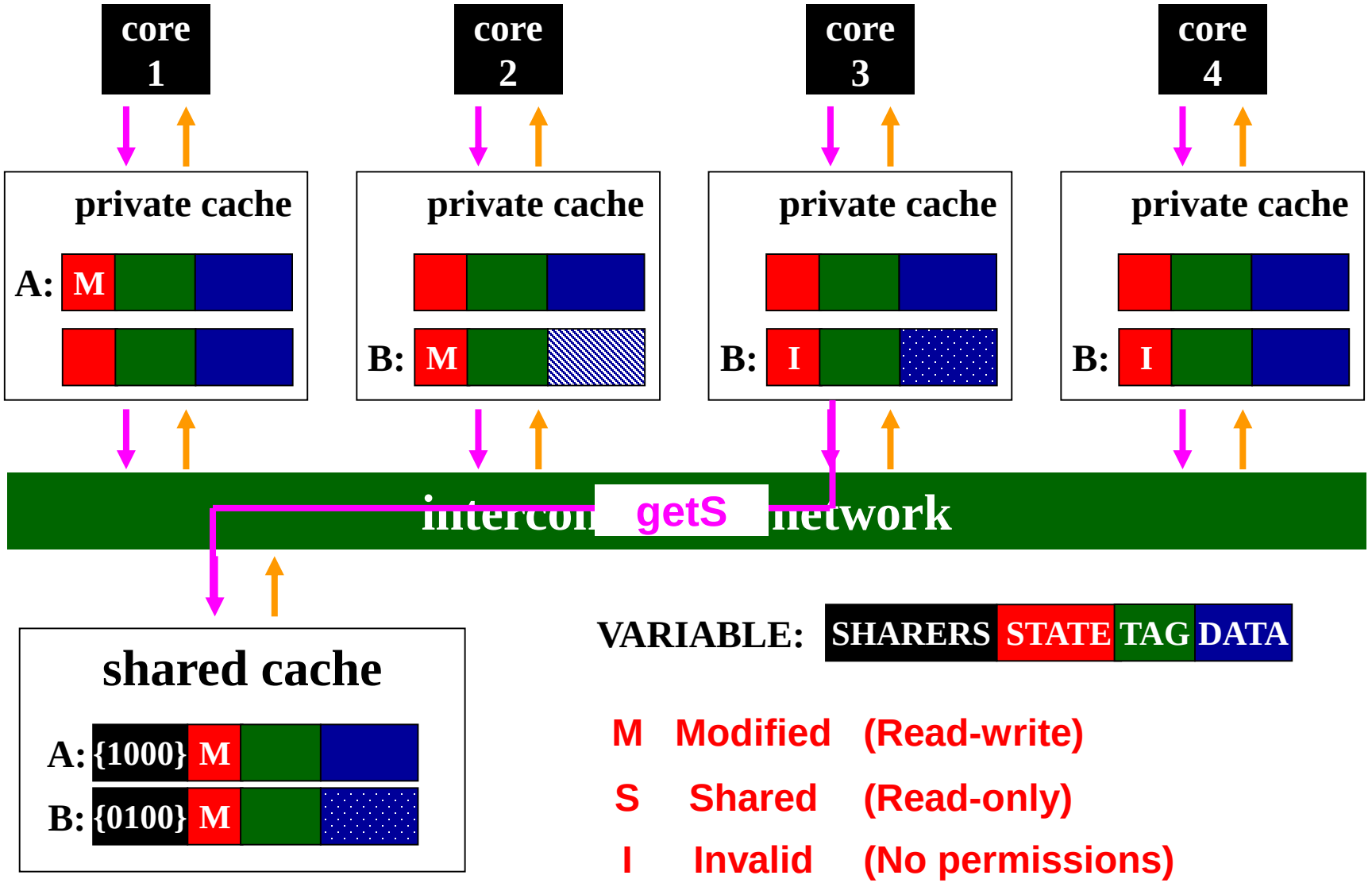
Shared cache concede permissão



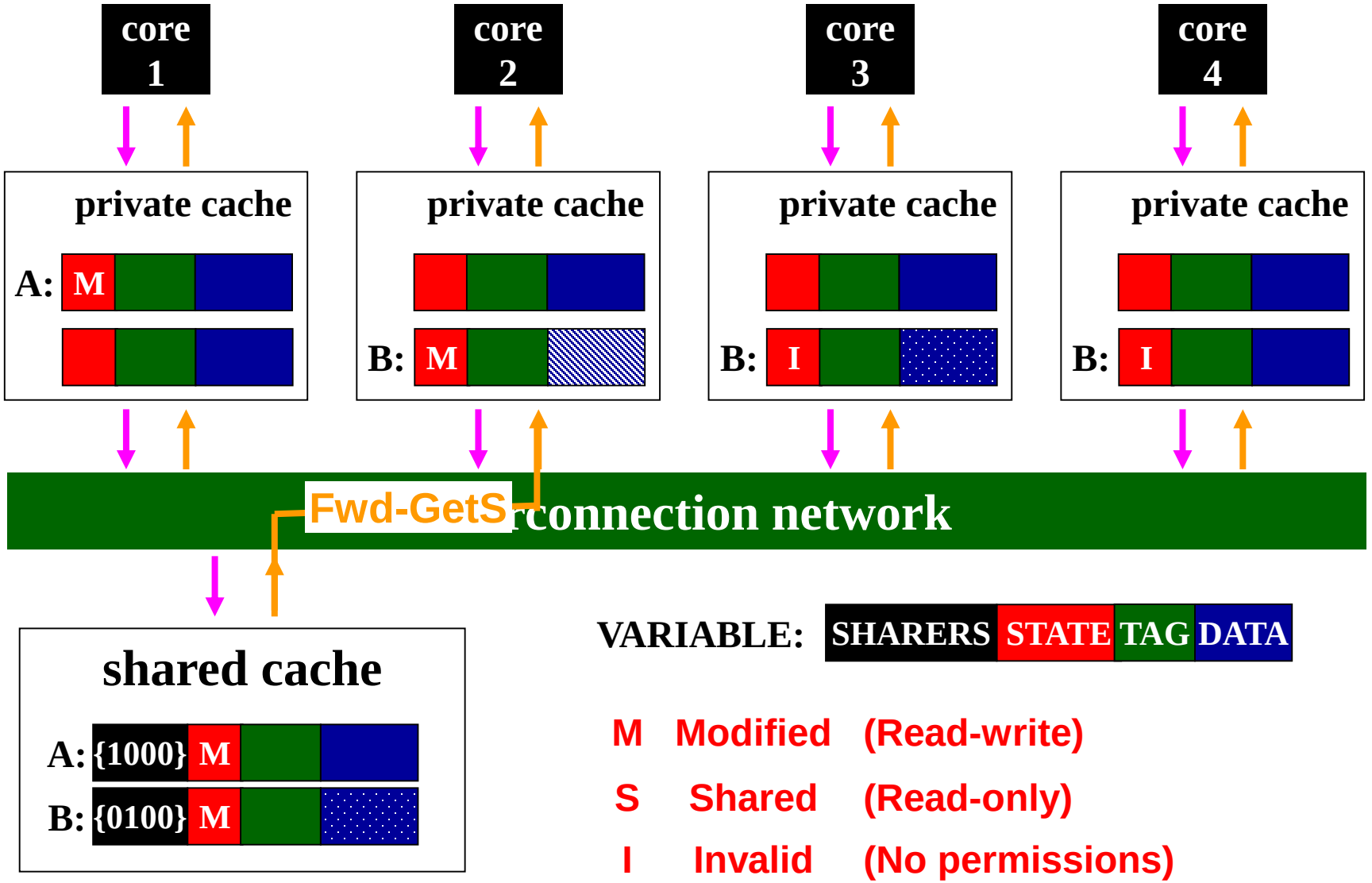
Shared cache concede permissão



Core 3 pede permissão para ler B



Shared cache encaminha pedido



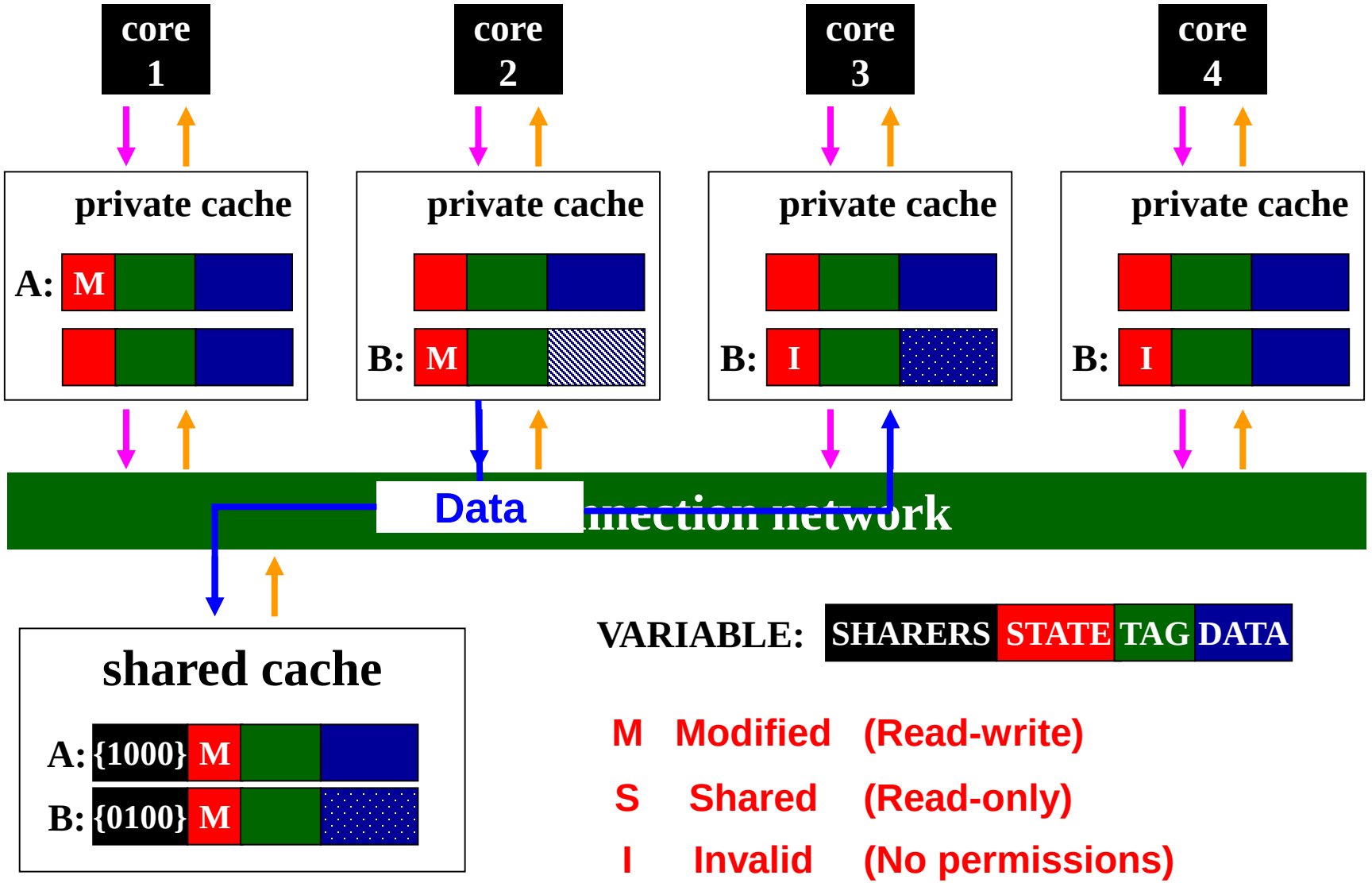
VARIABLE: SHARERS | STATE | TAG | DATA

M Modified (Read-write)

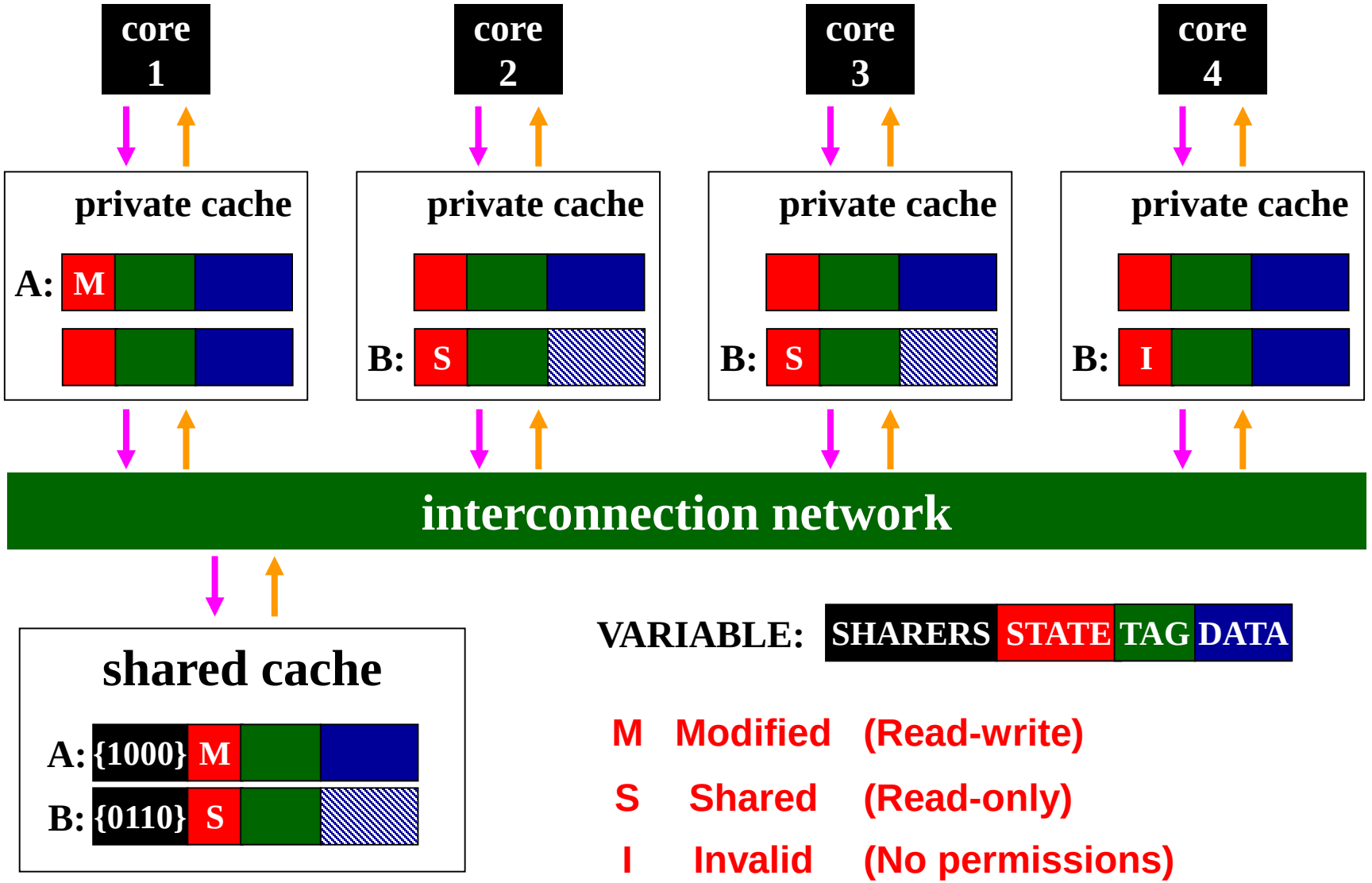
S Shared (Read-only)

I Invalid (No permissions)

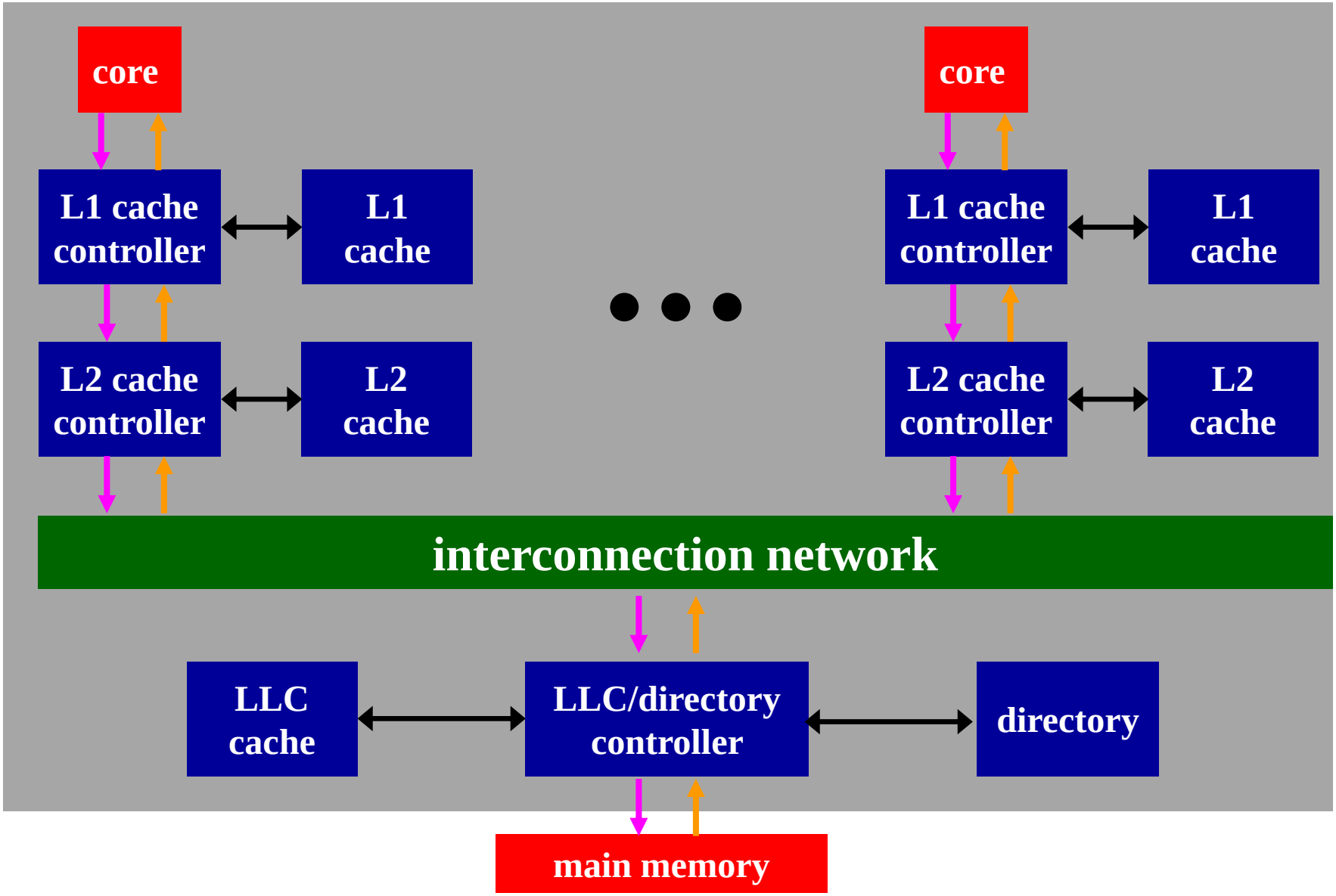
Private cache responde com dados



Core 3 observa valor atualizado



Estrutura de um *multicore chip*



Paralelismo e hierarquias de memória: coerência de cache