Cache Control and Cache Coherence Protocols

How to Manage State of Cache

How to Keep Processors Reading the Correct Information

Single Cache Control

- Direct mapped
- Write-back
- 4 word blocks
- 16 KB size, so 1024 blocks
- 32 bit addresses
- Valid & dirty bit for each block
  - 2-bit block offset, 2-bit byte offset
  - 10-bit cache address
  - 18-bit tag

Signals: Processor <-> Cache

1-bit read/write

1-bit request (for cache op)

32-bit address

32-bit data, processor to cache

32-bit data, cache to processor

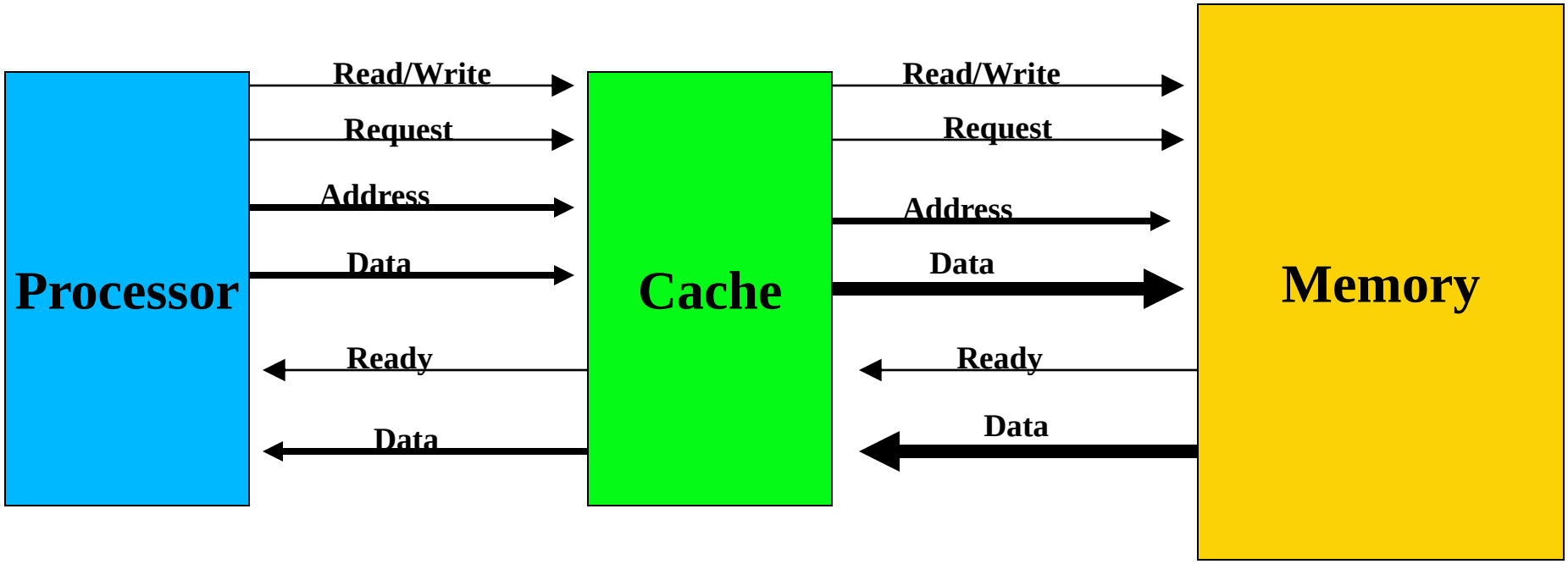
1-bit ready, indicating cache operation complete

If read, data is ready

Signals: Cache<-> Memory

- 1-bit read/write
- 1-bit request (for memory op)
- 32-bit address
- 128-bit data from cache to memory
- 128-bit data from memory to cache
- 1-bit ready, indicates memory operation complete
  - If read from memory, data is ready
  - If write, data is buffered and cache may proceed

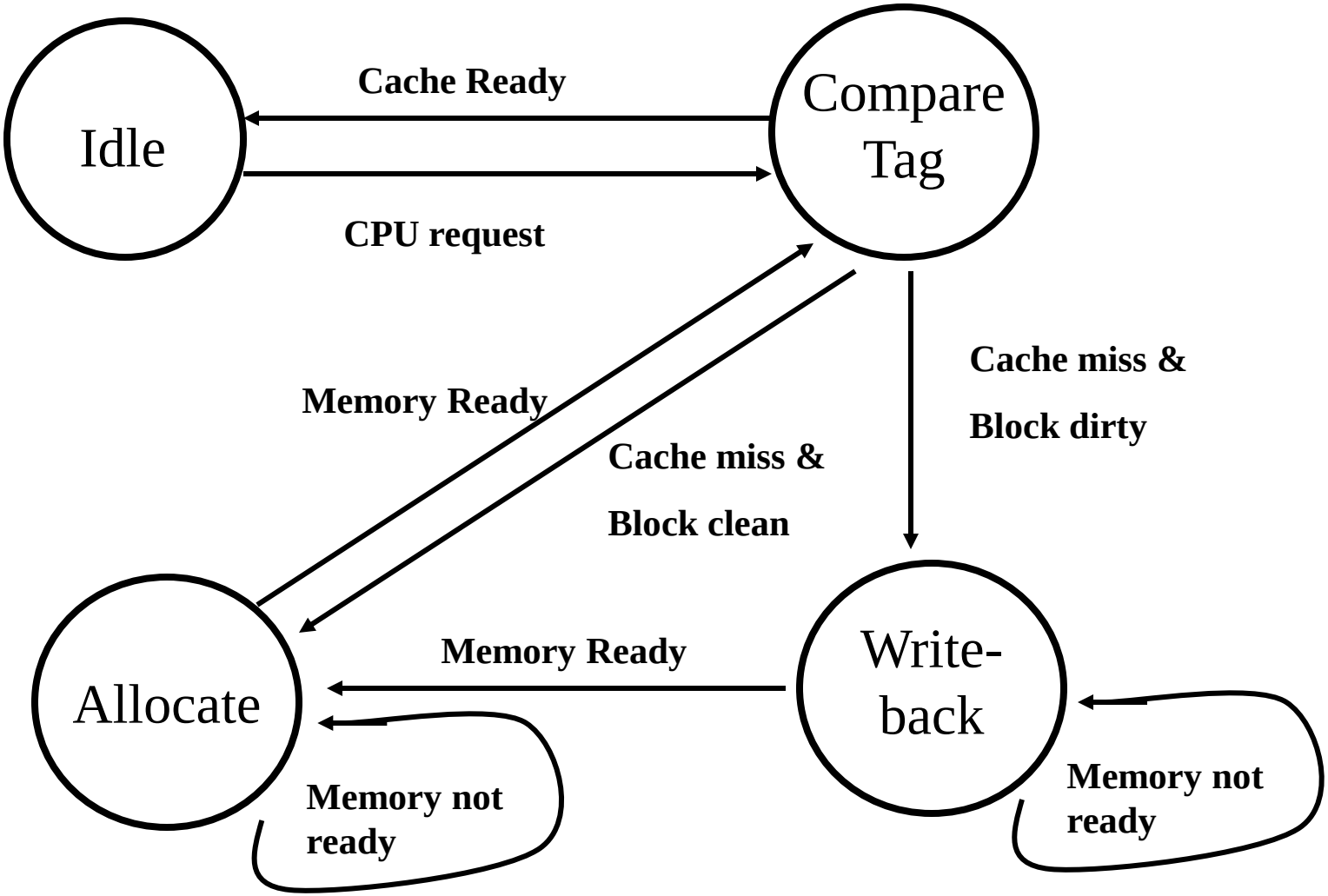
Communications pathways



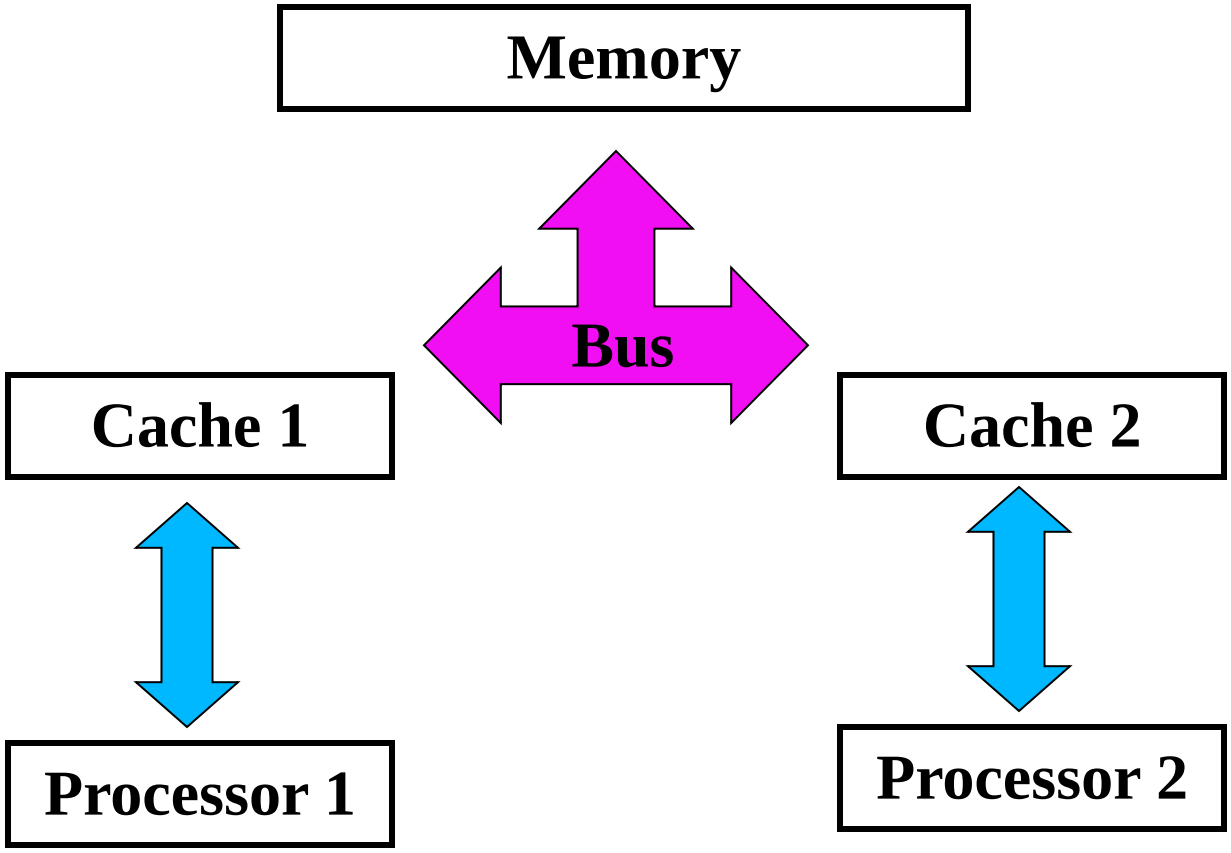
Cache States

- Idle – waiting for read/write request from processor
- Compare Tag
  - If valid & match (hit),
    - If also write, write data & set valid, set dirty
    - If not read, send data
    - Set cache ready
  - If not valid or not match (miss), set cache not ready
- Allocate
  - Read new block from memory
- Write-back
  - Write old block to memory

Cache controller FSM



Cache Coherence for multiple processors



Cache Coherence for multiple processors

- Reads
  - Several Processors can store the same information in their own caches for reading without problems
- Writes
  - Other processors will no longer have the correct information in their caches
  - Write Back protocol to keep bus traffic lower

Cache ***Block*** States

- Uses bits to indicate appropriate use of cache line/block
- Depends on actions of the given processor and other processors
- Keep track using Finite State Machine Model -- for each line/block

Cache State

- How can cache for one processor change state based on actions of another processor?
  - "Snoopy" bus protocol
  - each cache uses bus to access memory, possibly to signal other processors
  - each cache "snoops" on the bus, watching actions of other processors which may affect its own state, eaves-dropping
  - a cache may change state based on action of other processors on bus, without special signals

Finite State Machine representation

State 1: Invalid

- information in this cache block is invalid, regardless of the tags

State 2: Read only (clean, shared)

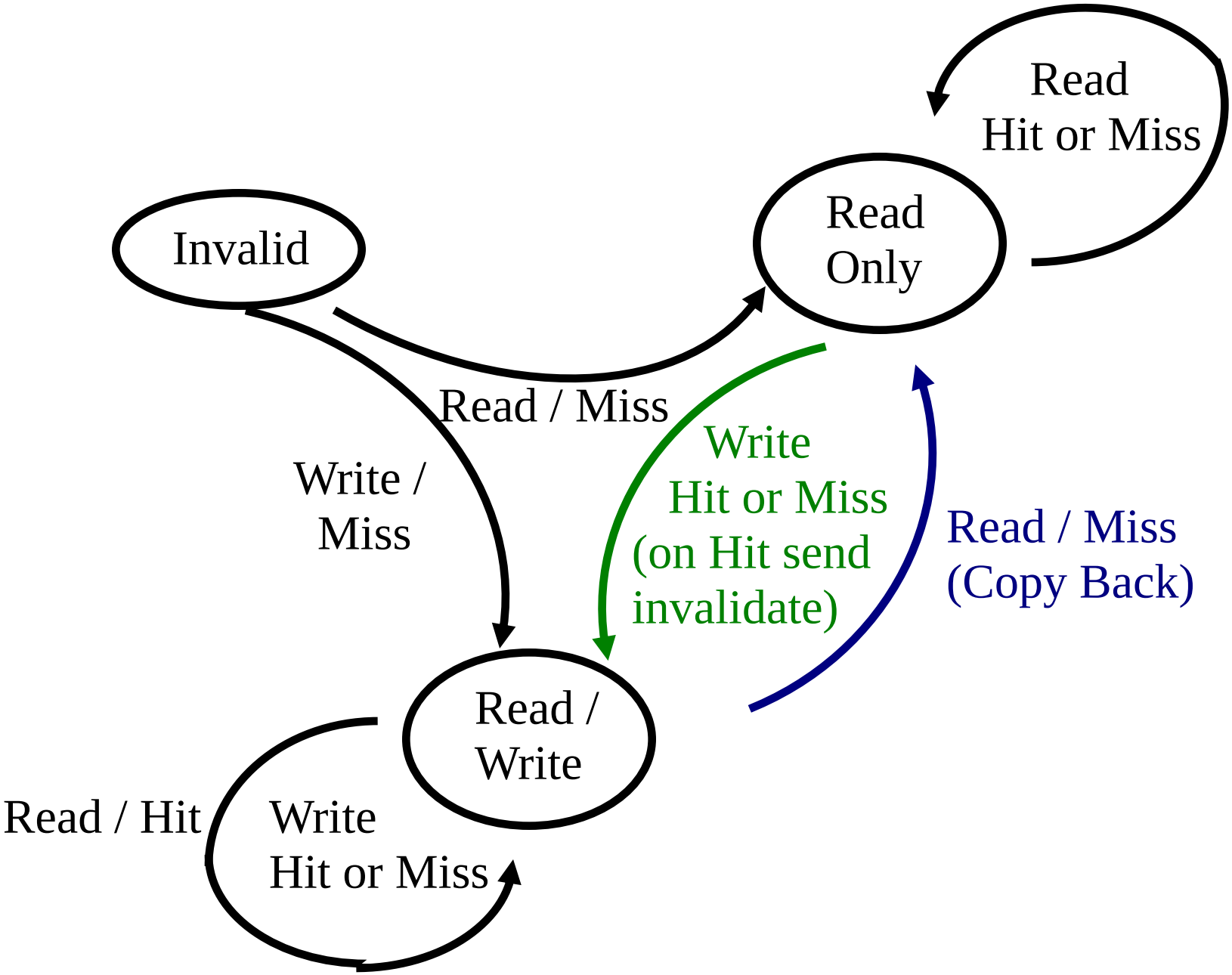
- information in this cache block may be read locally, without any bus traffic

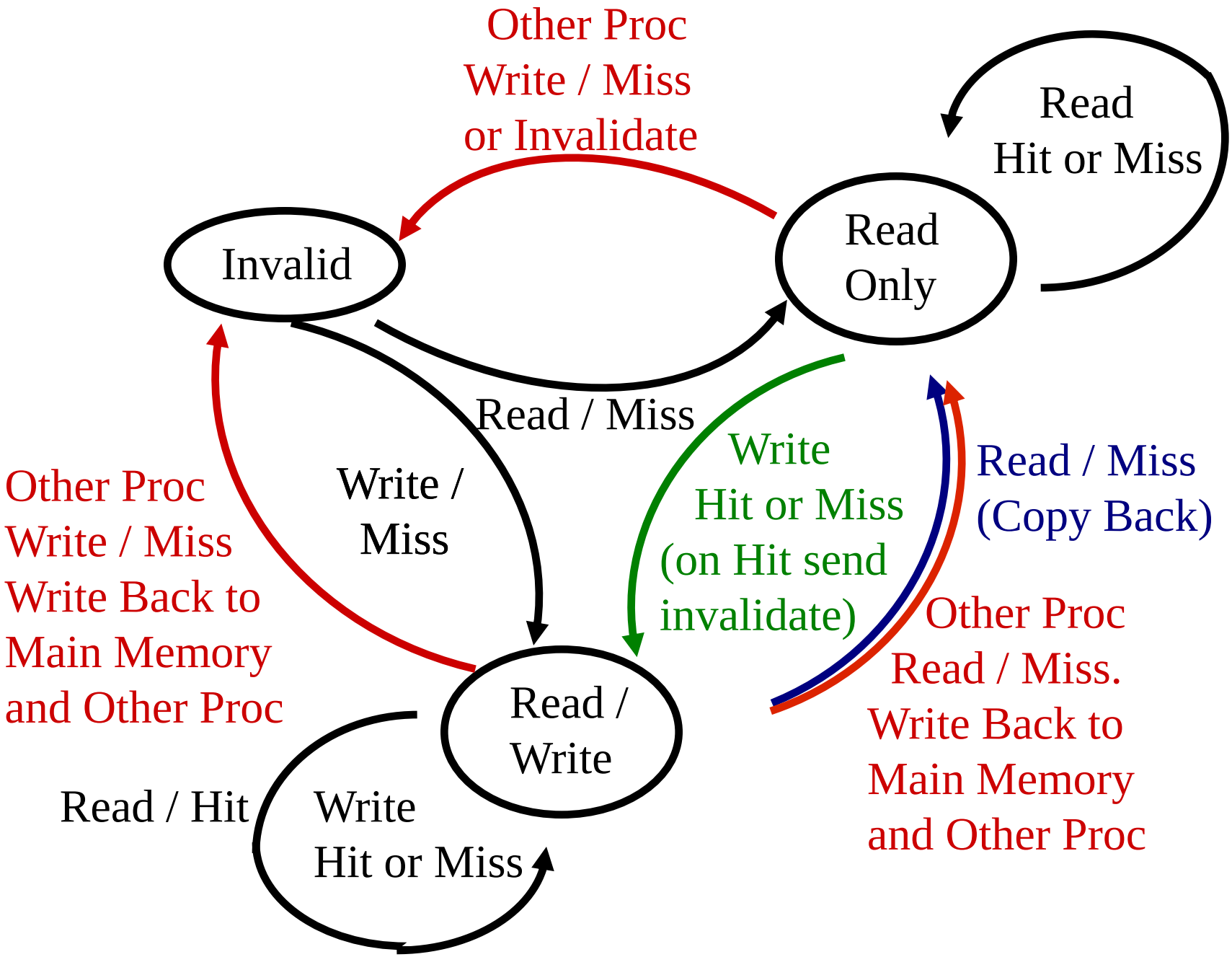
State 3: Read/Write (dirty, exclusive)

- information in this cache block may be read or written locally, without any bus traffic
- information must be copied back to main memory when cache block is replaced or when block is used by another processor

Finite State Machine representation

- State Changes based on local processor action
- State Changes based on other processor action
  - Depends on seeing action on the bus, e.g. misses or an explicit signal
  - Sees address (cache-block address and tag) on the bus to know if action applies to this cache block
- If Dirty and State changes, must Copy Back to Main Memory
- Read Only = Clean = Shared
- Read / Write = Dirty


Read
Hit or Miss
Read
Only
Invalid
Read / Miss
Write /
Miss
Write
Hit or Miss
(on Hit send
invalidate)
Read / Miss
(Copy Back)
Read /
Write
Read / Hit
Write
Hit or Miss


Other Proc
Write / Miss
or Invalidate
Read
Hit or Miss
Invalid
Read
Only
Read / Miss
Write /
Miss
Write
Hit or Miss
(on Hit send
invalidate)
Read / Miss
(Copy Back)
Other Proc
Write / Miss
Write Back to
Main Memory
and Other Proc
Read /
Write
Other Proc
Read / Miss.
Write Back to
Main Memory
and Other Proc
Read / Hit
Write
Hit or Miss

Cache Control FSM