ECE 2300
Digital Logic & Computer Organization

Fall 2016

Caches

Cornell University

Course Content

- Binary numbers and logic gates
- Boolean algebra and combinational logic
- Sequential logic and state machines

cut off for Prelim 1

start of Prelim 2

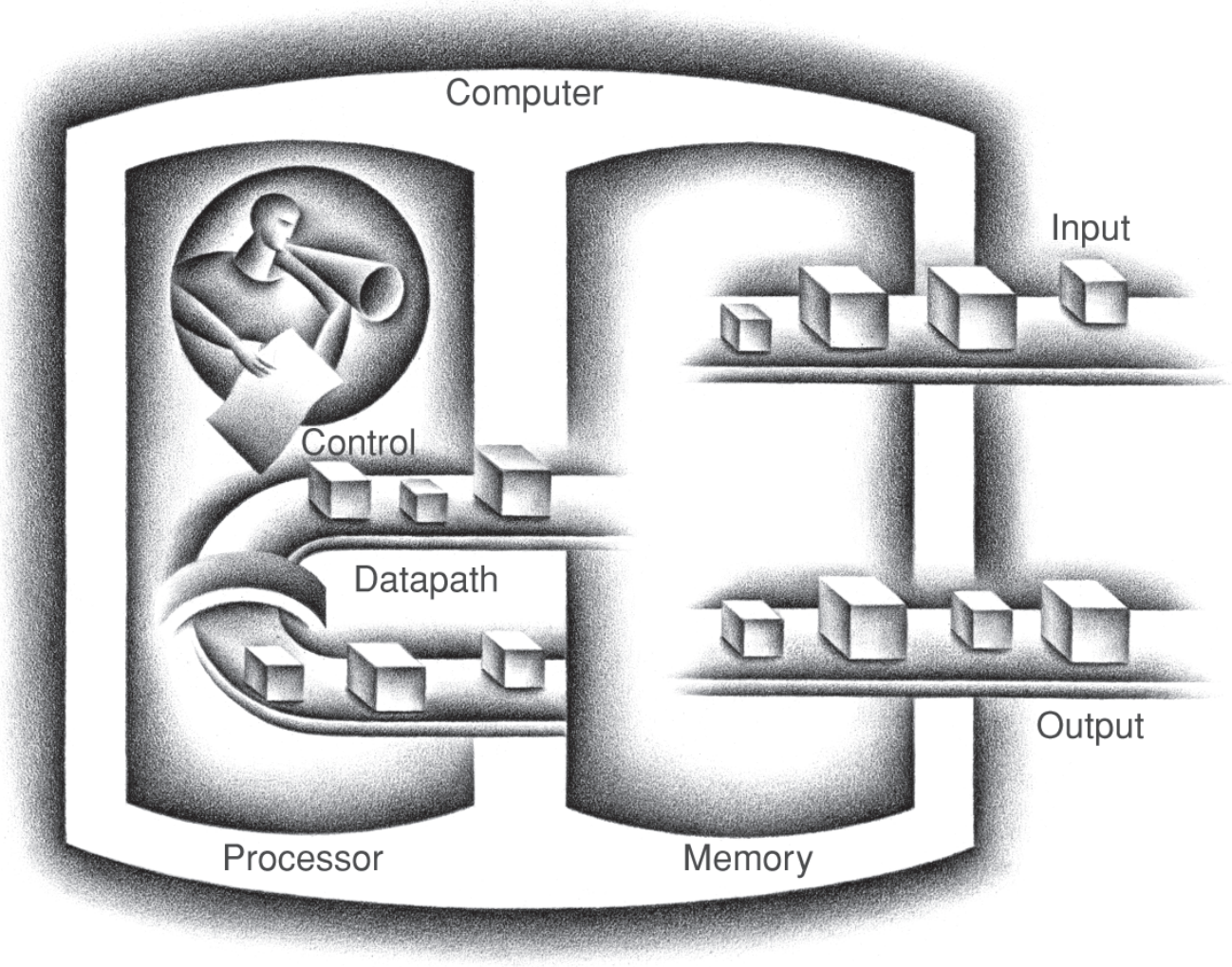
- Binary arithmetic
- Memories

- Instruction set architecture
- Processor organization

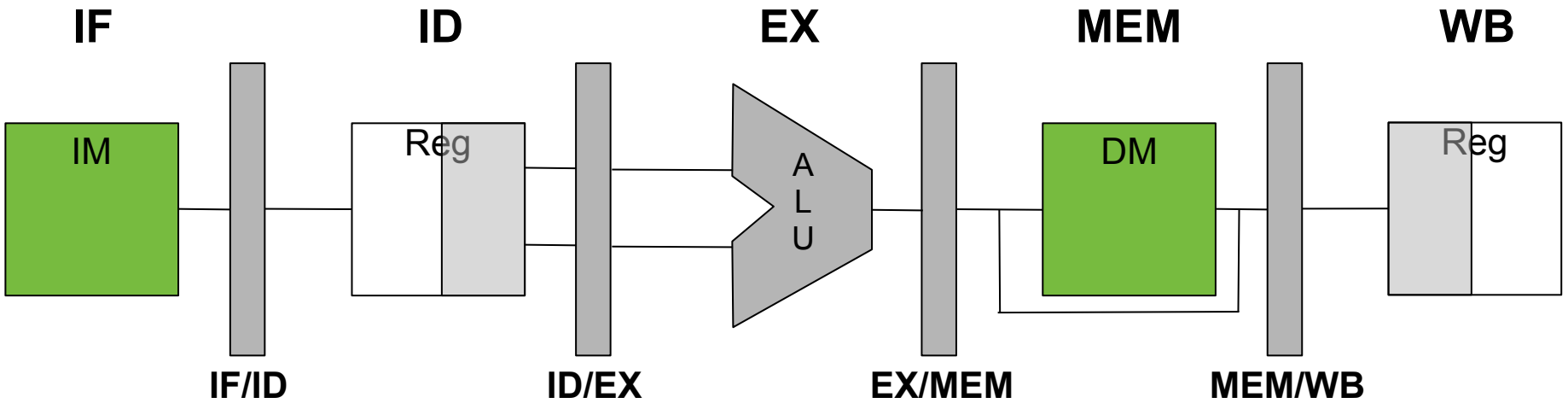
cut off for Prelim 2

- Caches and virtual memory
- Input/output
- Case studies

Organization of a Computer



We Need Very Fast Memory



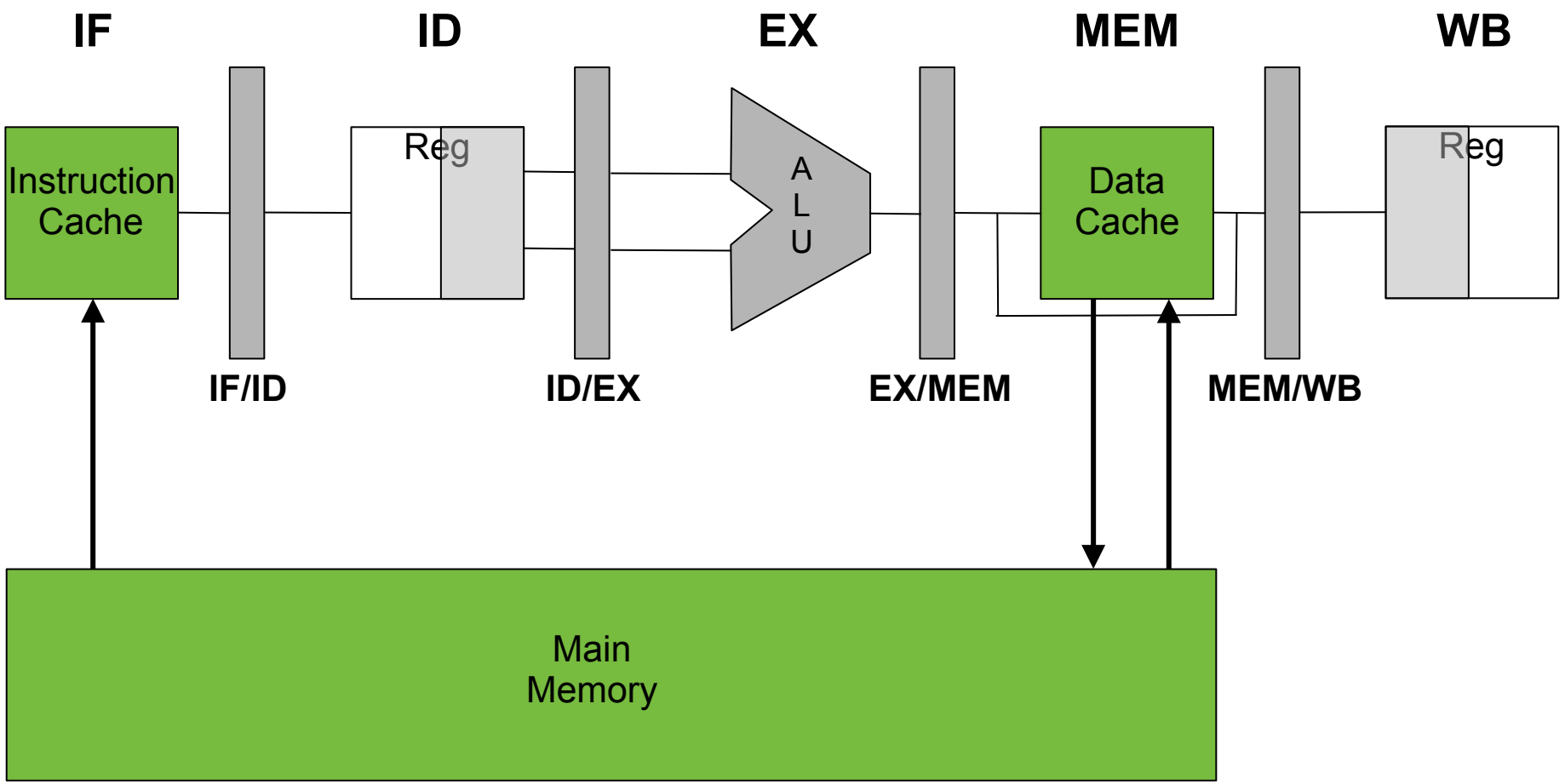
Memory Technology

- Processor cycle time ~250ps-2ns (4GHz-500MHz)

- DRAM (Dynamic Random Access Memory)
  - Slow: 10's of ns for a read or write (1 ns = 1000ps)
  - Small storage cell: 1 transistor + capacitor → GBs/chip (gigabytes)
  - Inexpensive: 10¢/MB (megabyte)
  - Used for *main memory*

- SRAM (Static Random Access Memory)
  - Fast: 100's of ps to few ns for a read or write
  - Large storage cell: 6 transistors
  - Expensive: $1/MB
  - Used for *caches*

Using *Caches* in the Pipeline



Cache

- Small SRAM memory that permits rapid access to a subset of instructions or data
- If the data is in the cache (*cache hit*), we retrieve it without slowing down the pipeline
- If the data is not in the cache (*cache miss*), we retrieve it from the main memory (DRAM)
- The *hit rate* is the fraction of memory accesses found in the cache
  - The *miss rate* is *1-hit rate*

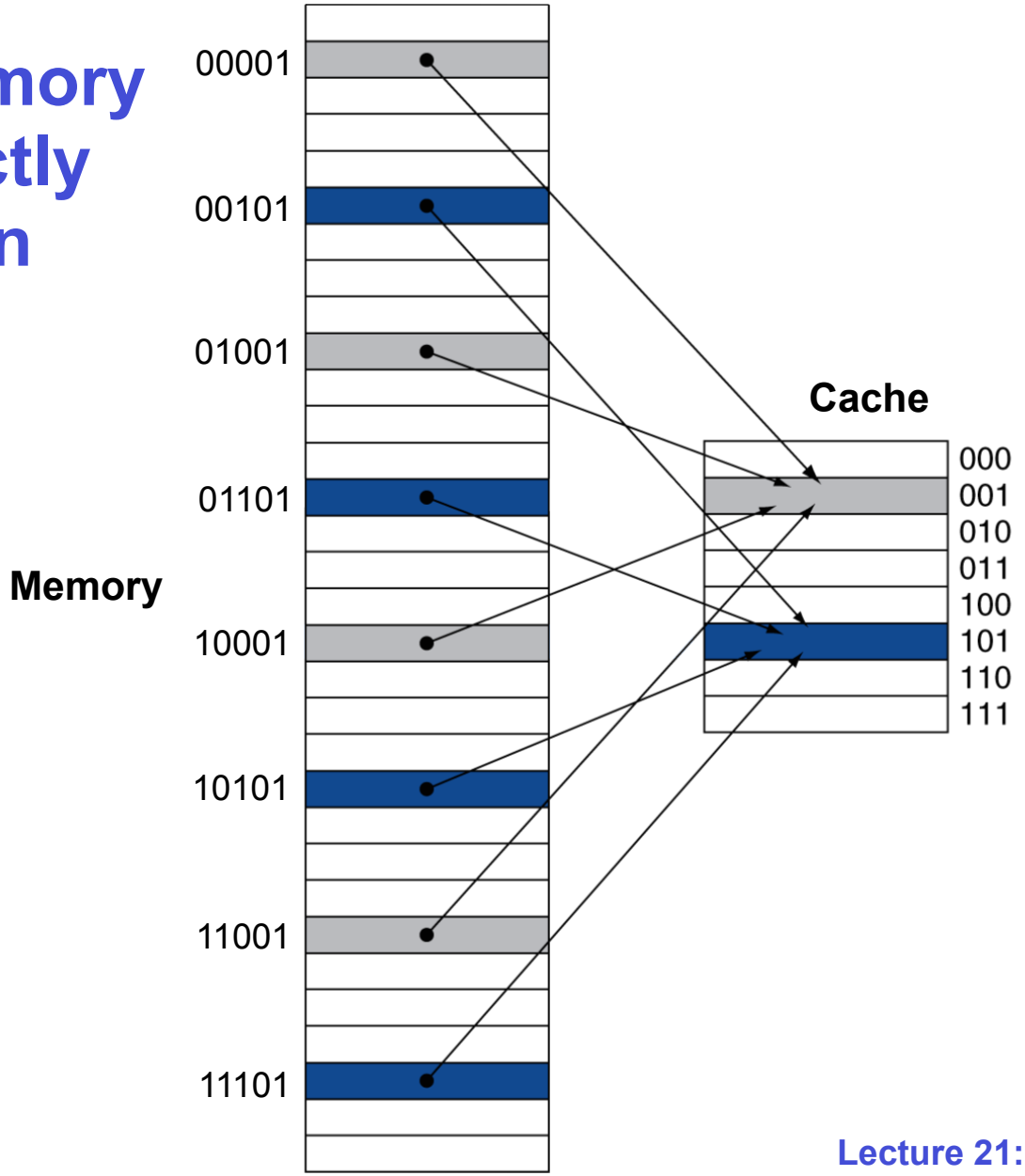
Why Caches Work: Principle of Locality

- **Temporal locality**
  - If memory location X is accessed, then it is likely to be accessed again in the near future
  - Caches exploit temporal locality by keeping a referenced instruction or data in the cache

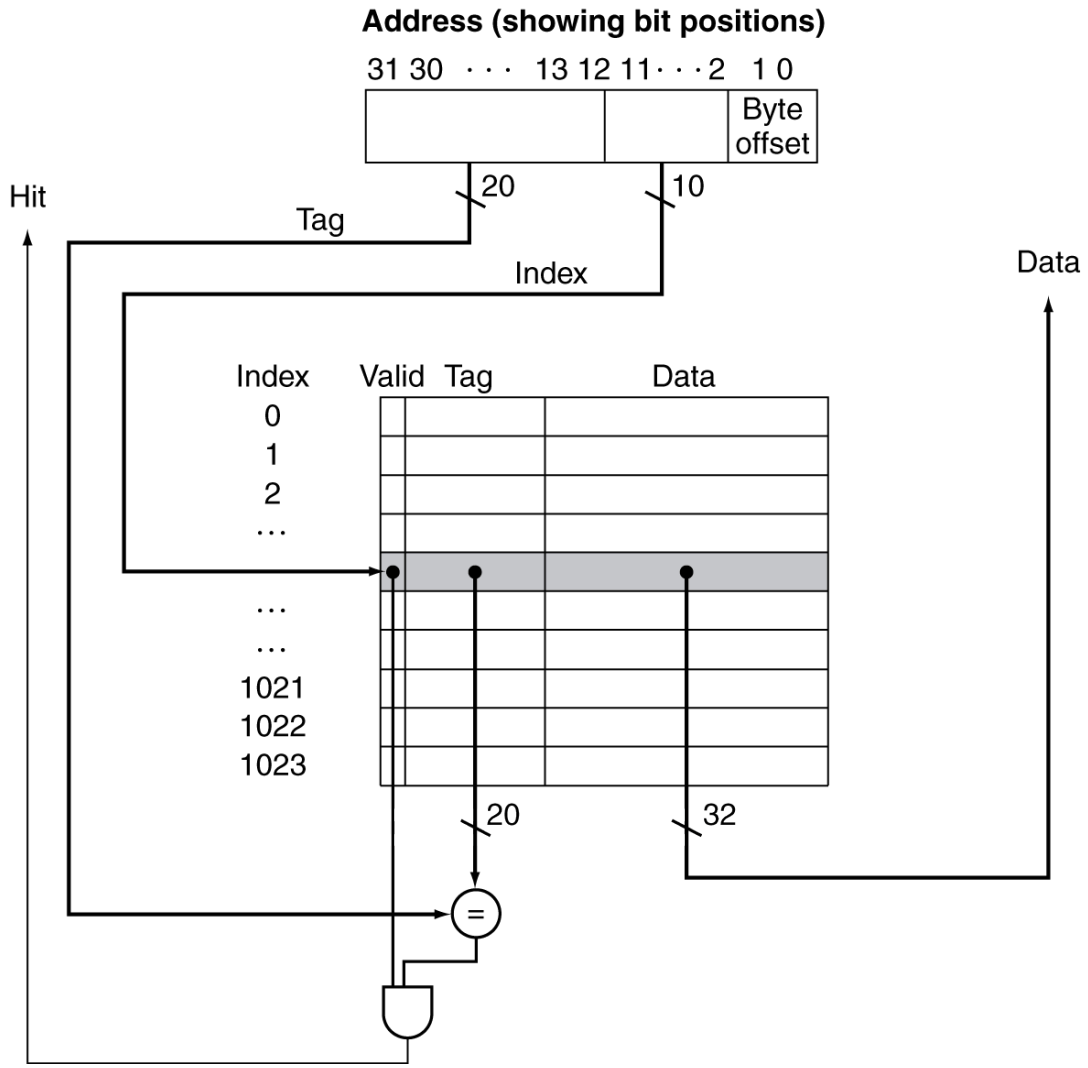
- **Spatial locality**
  - If memory location X is accessed, then locations near X are likely to be accessed in the near future
  - Caches exploit spatial locality by bringing in a *block* of instructions or data into the cache on a miss

Direct Mapped Cache

- Each block in memory is mapped to exactly one cache location

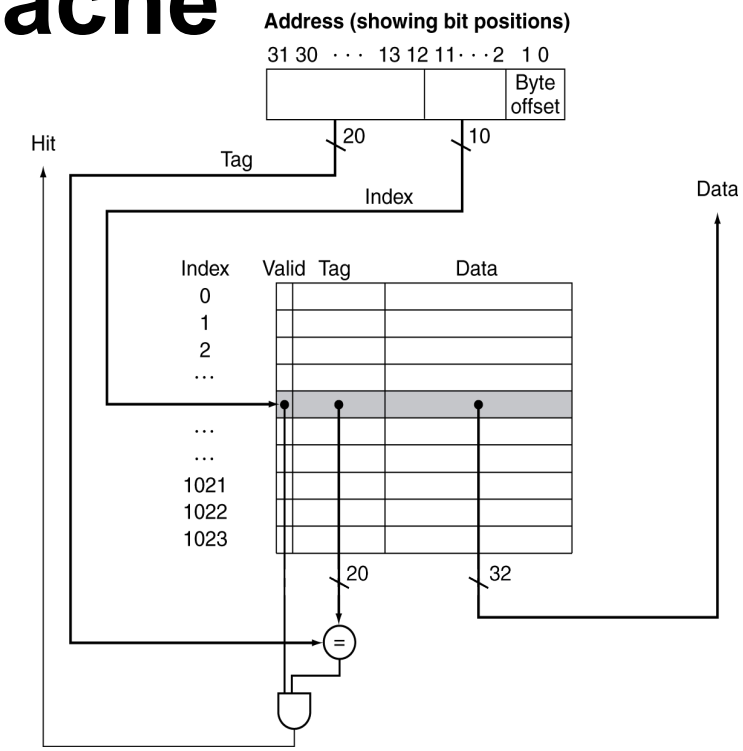


Direct Mapped Cache



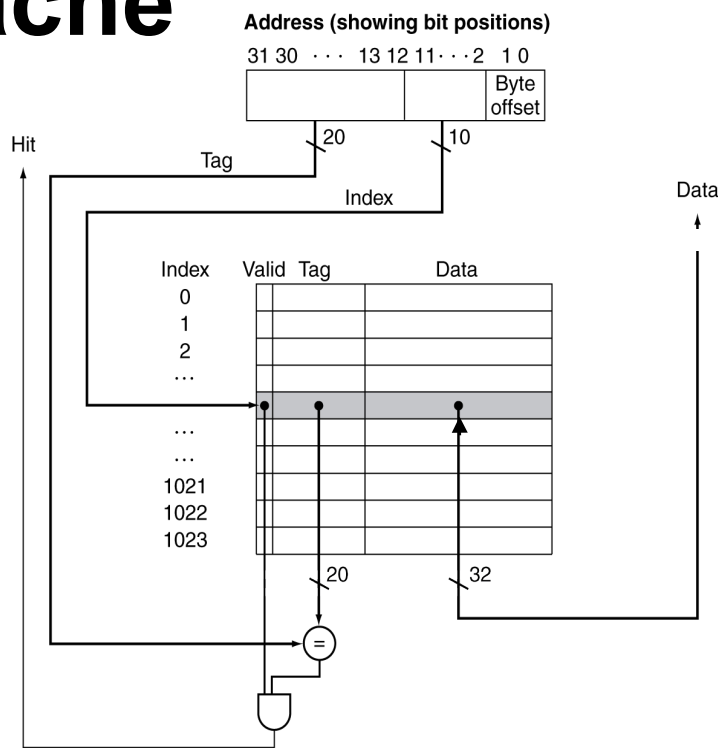
Reading DM Cache

- Use the index bits to retrieve the tag and data
- Compare the tag from the address with the retrieved tag
- If a match (hit), select the desired data using the byte offset
- If a mismatch (miss)
  - Bring the block from memory into the cache
  - Store the tag from the address with the block
  - Select the desired data using the byte offset



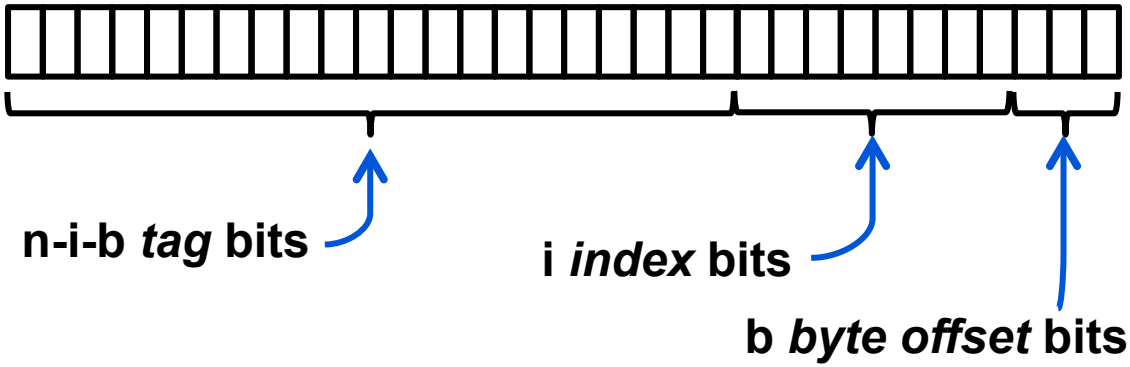
Writing DM Cache

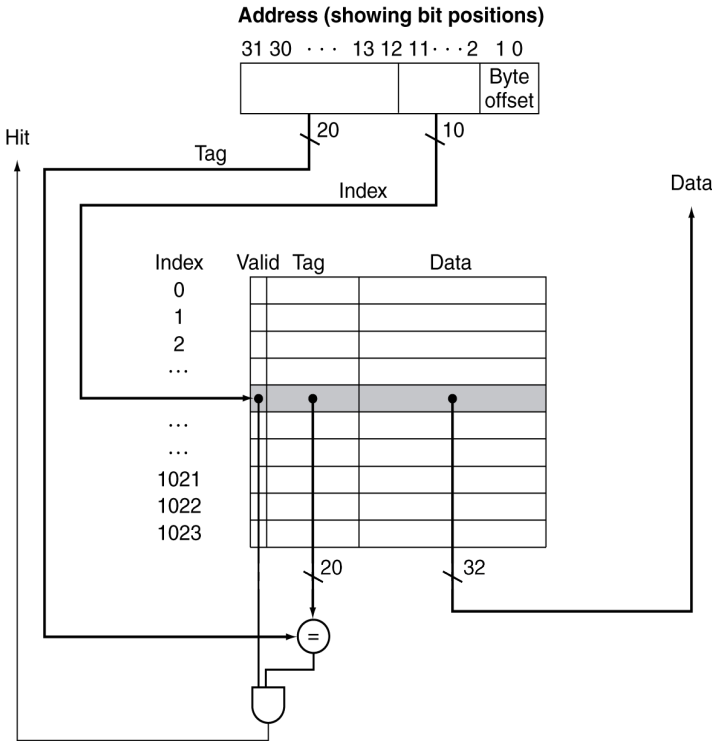
- Use the index bits to retrieve the tag
- Compare the tag from the address with the retrieved tag
- If a match (hit), write the data into the cache location
- If a mismatch (miss), one option
  - Bring the block from memory into the cache
  - Store the tag from the address with the block
  - Write the data into the cache location



Direct Mapped Cache

- **Memory address break down**

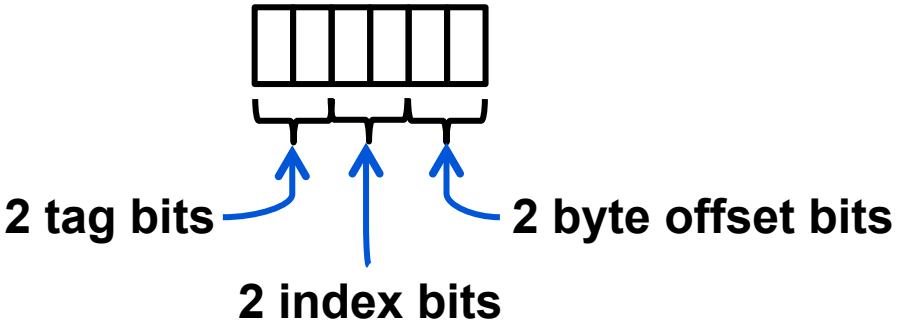


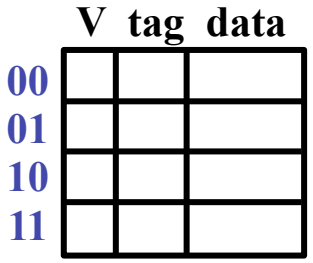


- **Cache parameters**
  - Size of each block is 2^b bytes
  - Number of blocks is 2^i
  - Total cache size is 2^{b+i} bytes

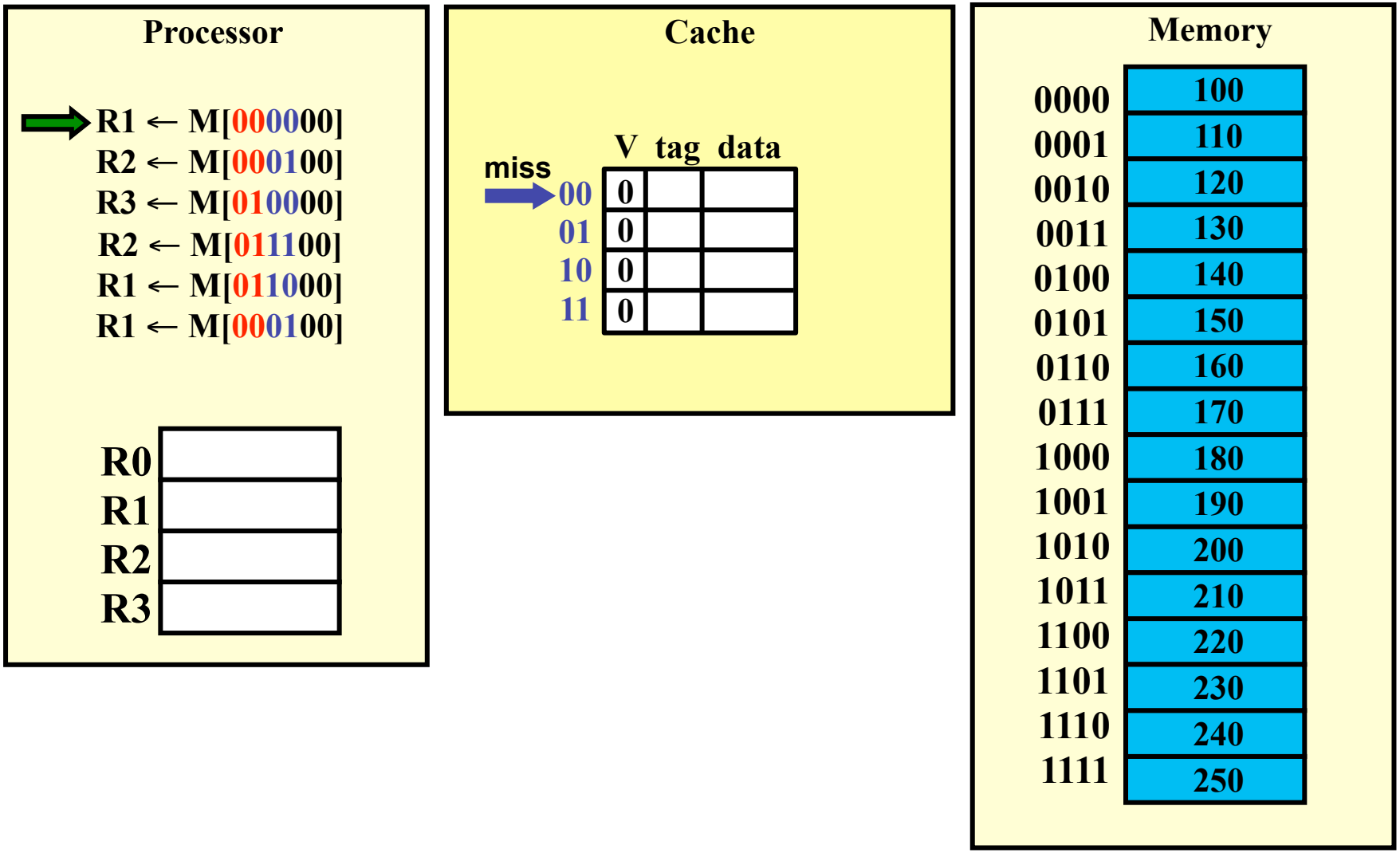
Direct Mapped Data Cache Example

- Size of each block is 4 bytes
- Cache holds 4 blocks
- Memory holds 16 blocks
- Memory address

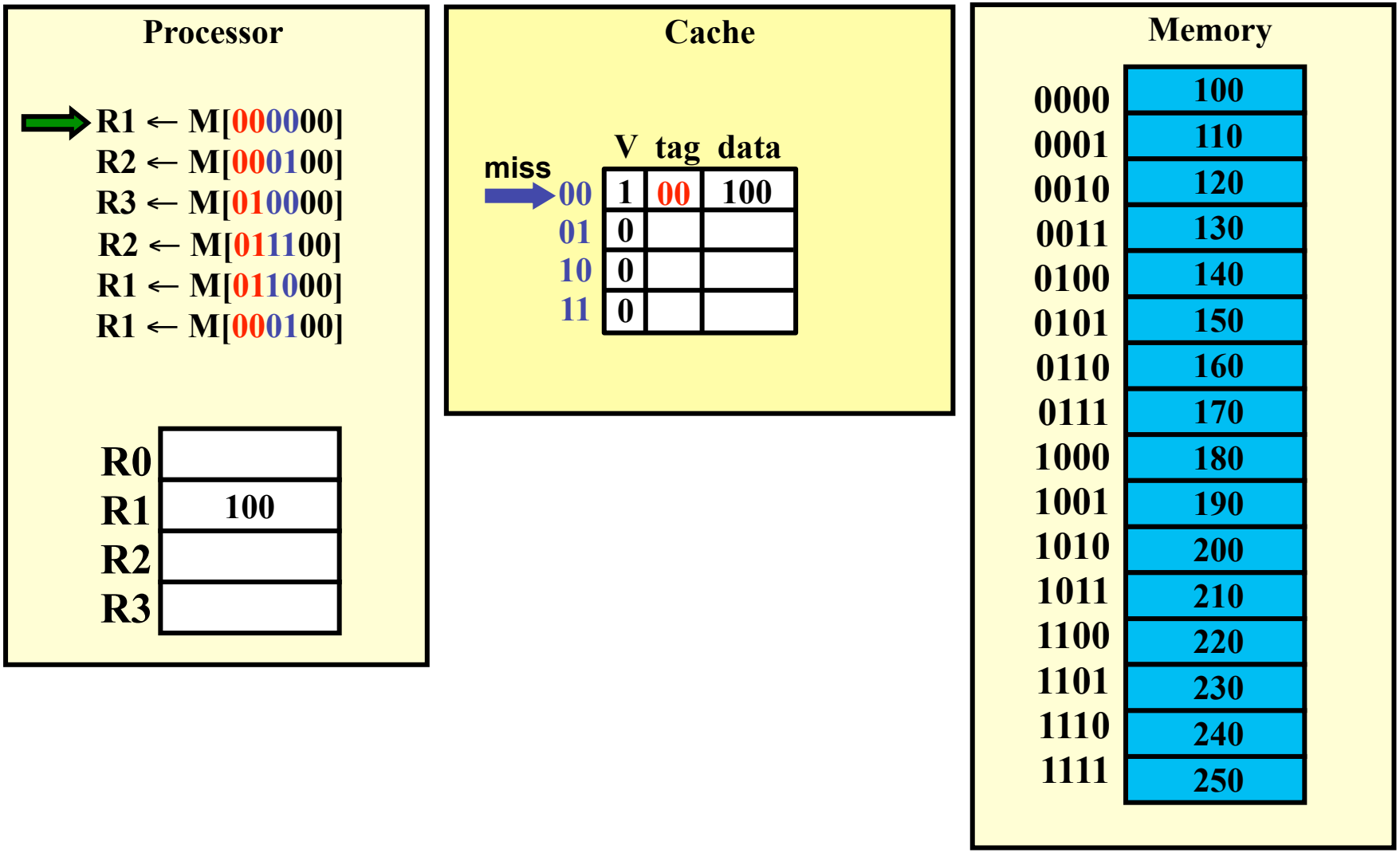




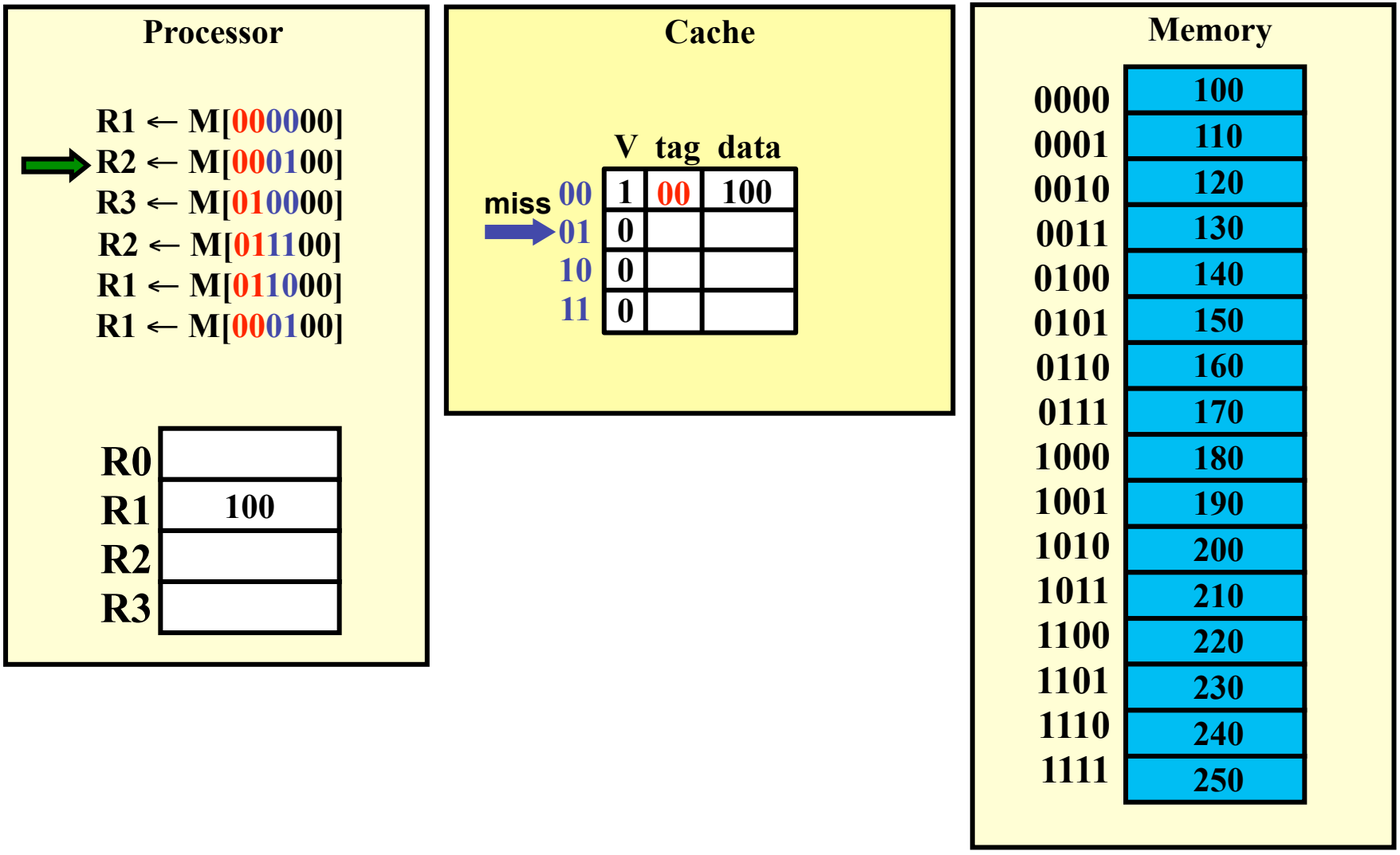
Direct Mapped Data Cache Example



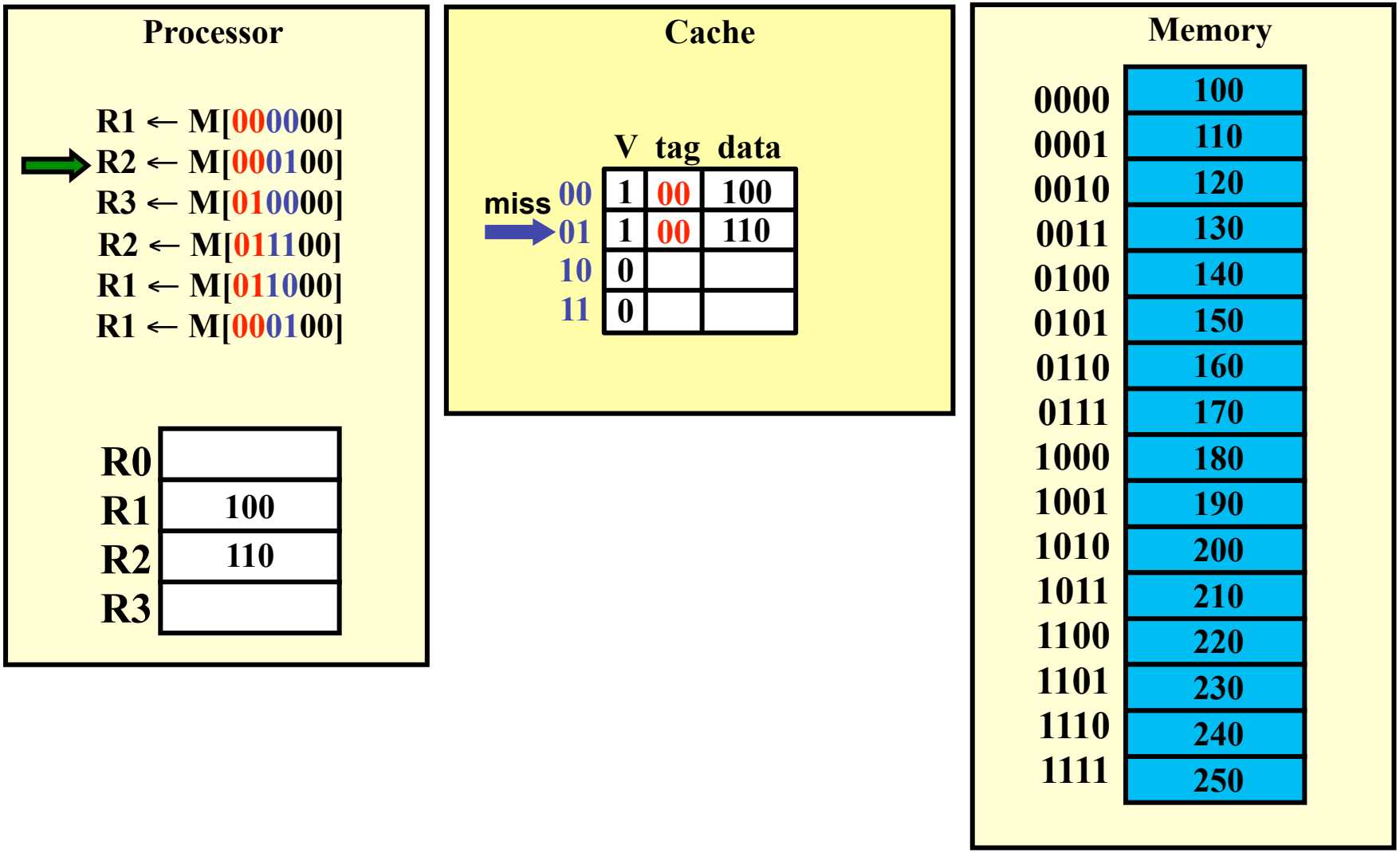
Direct Mapped Data Cache Example



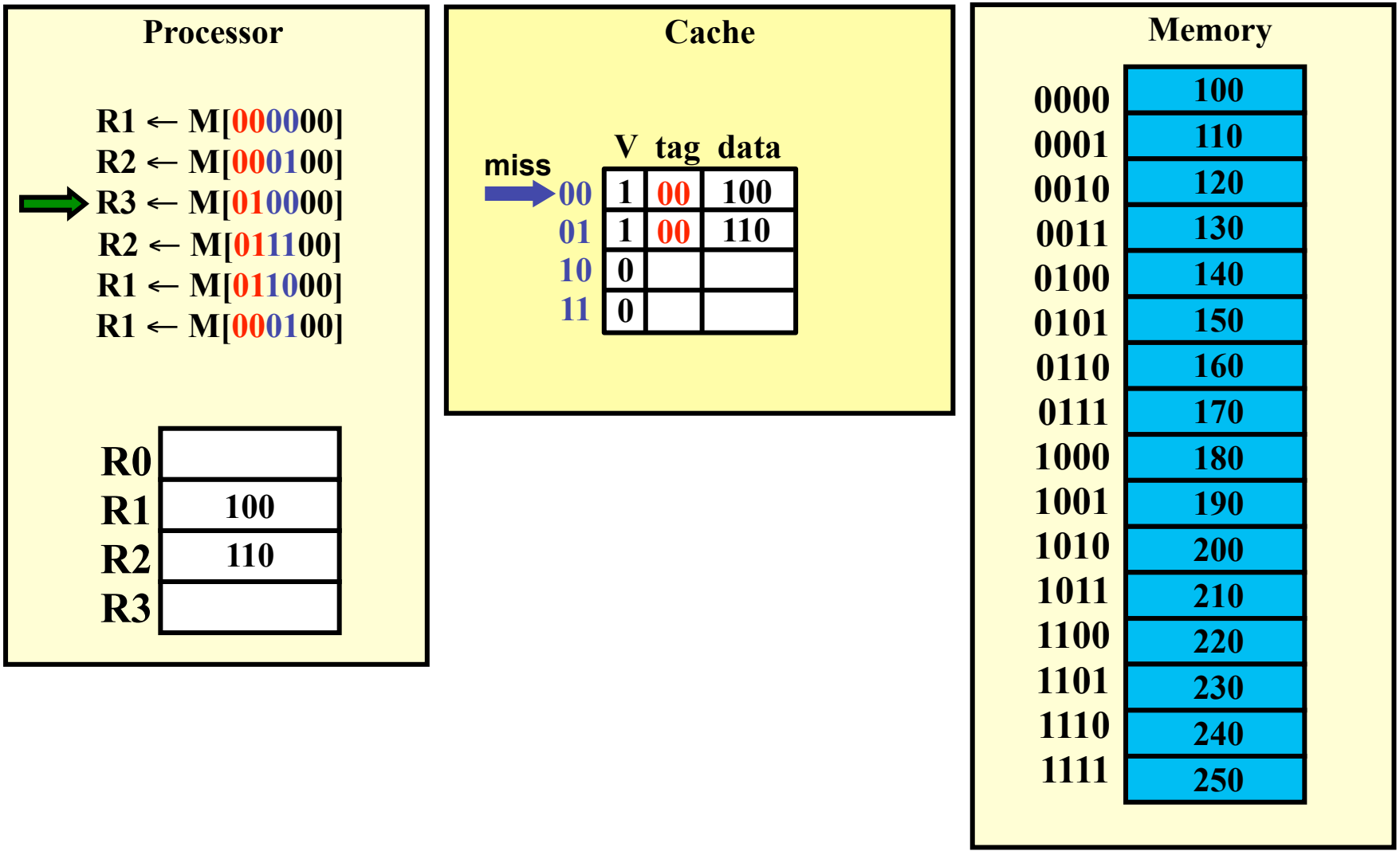
Direct Mapped Data Cache Example



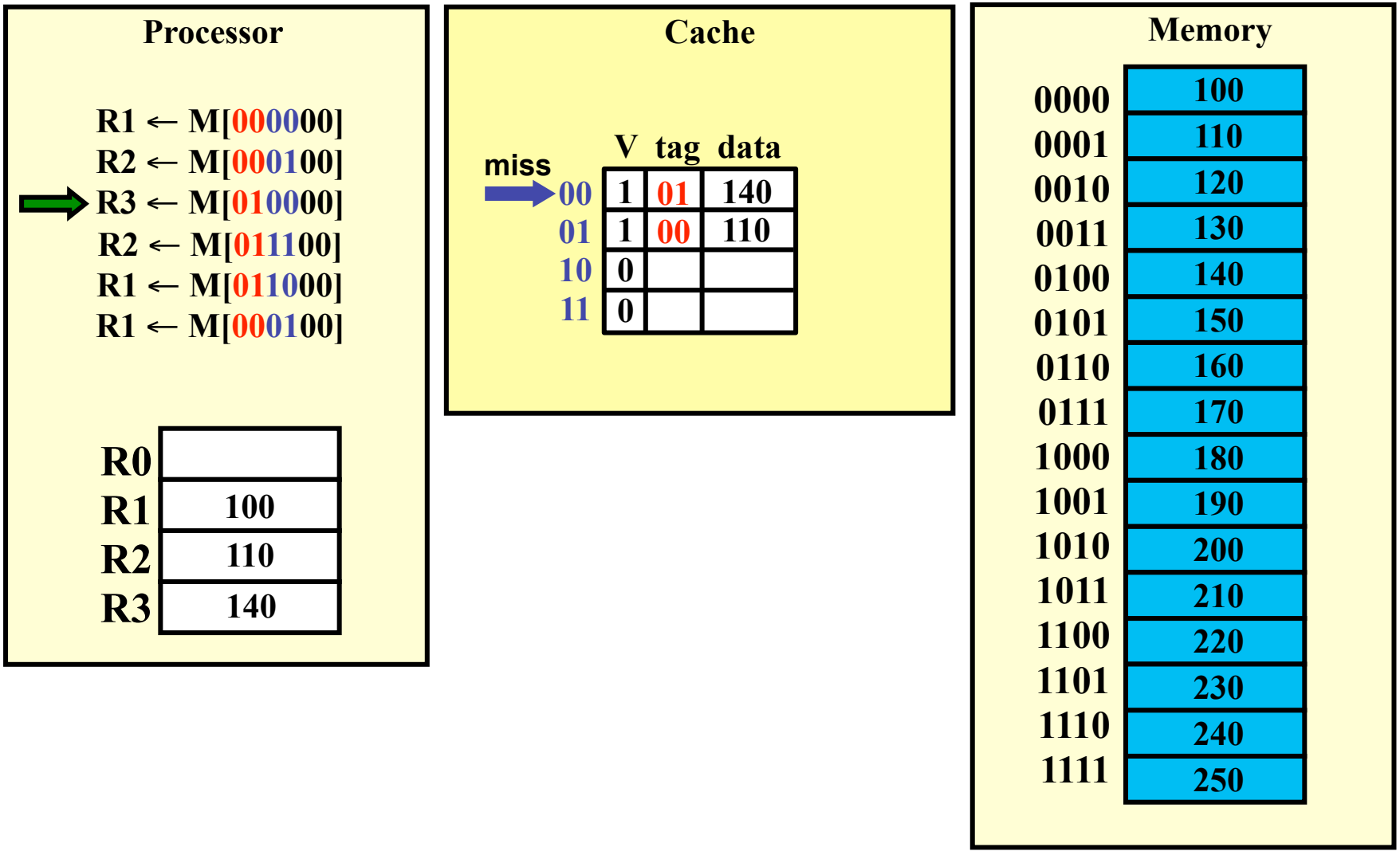
Direct Mapped Data Cache Example



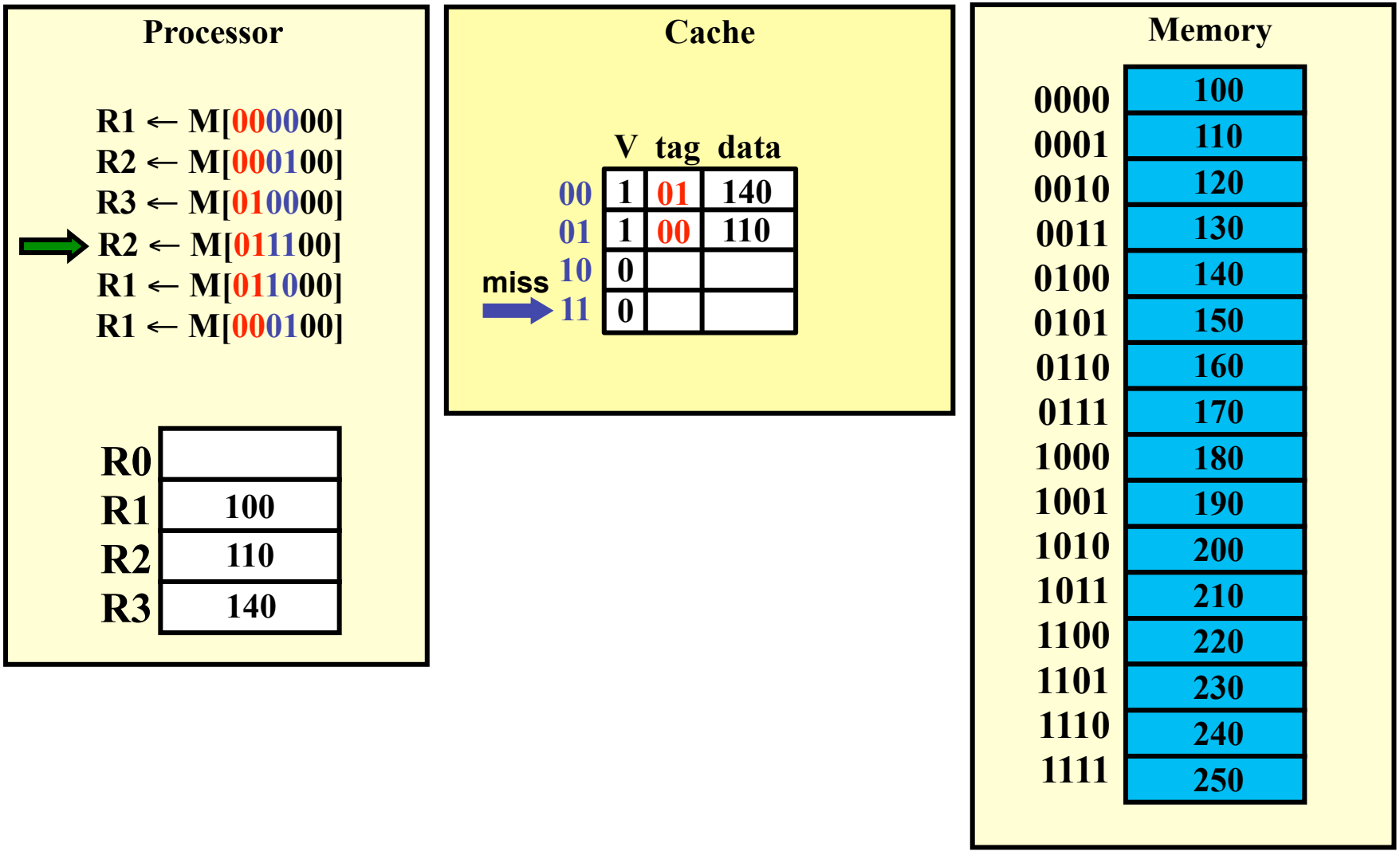
Direct Mapped Data Cache Example



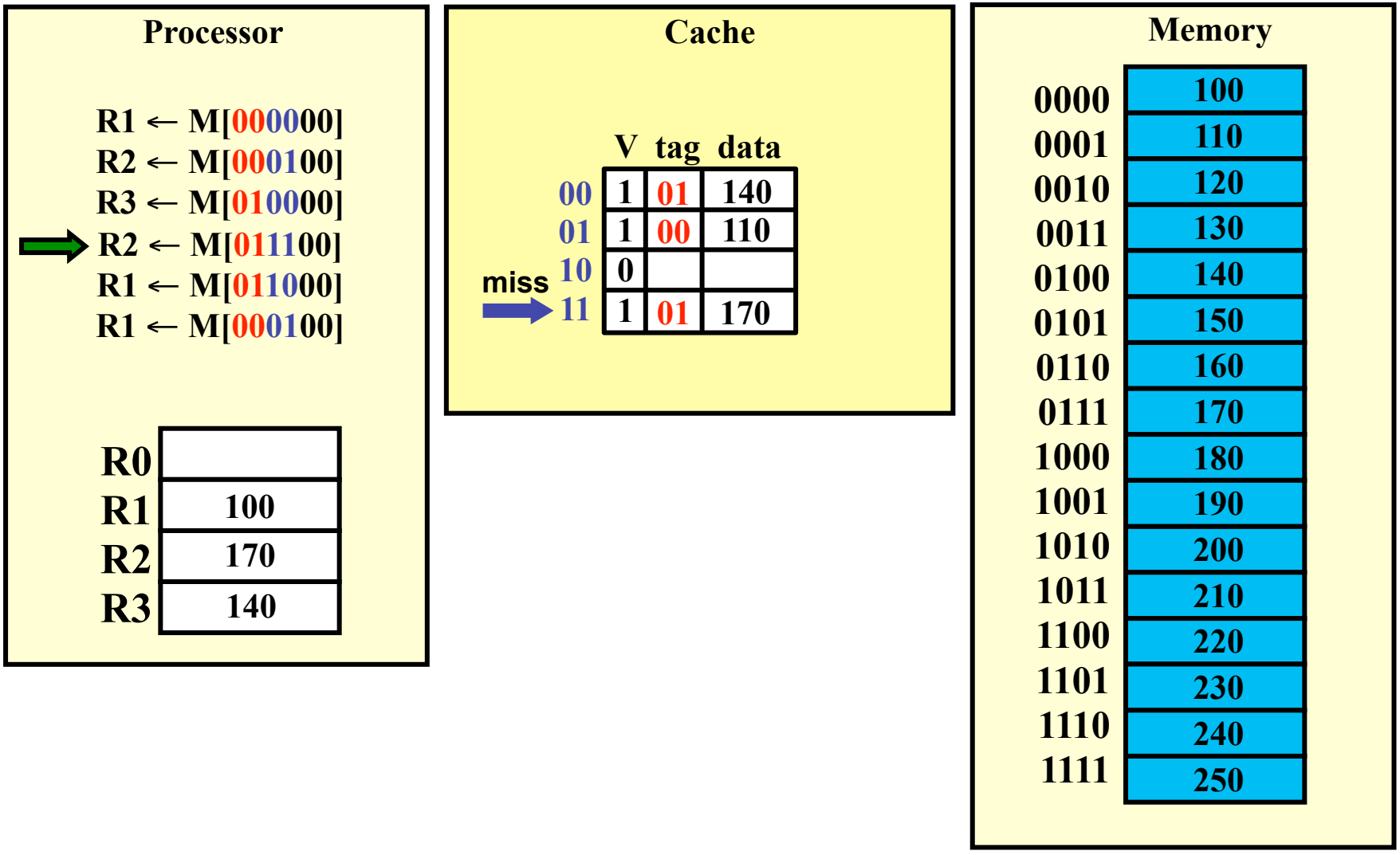
Direct Mapped Data Cache Example



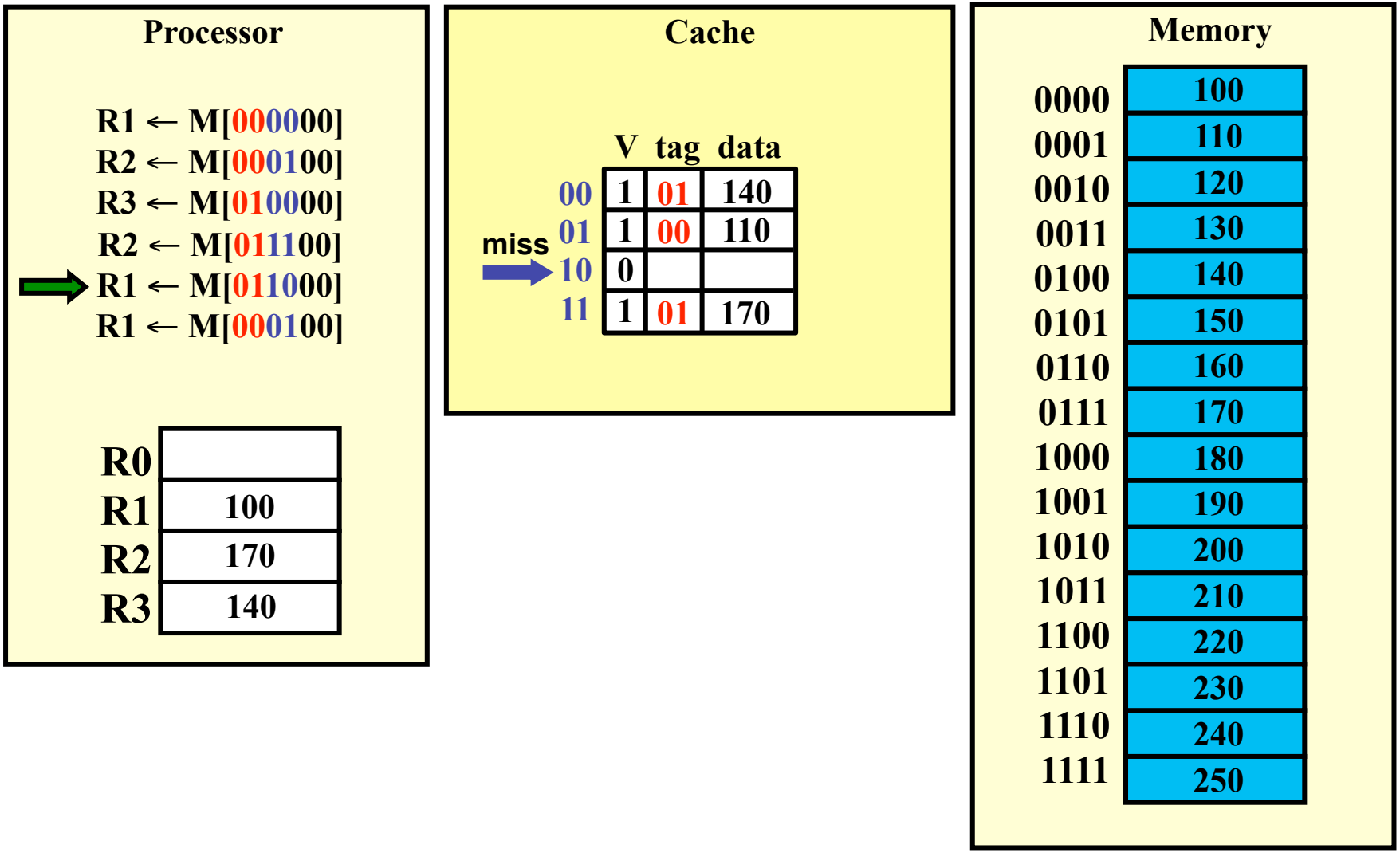
Direct Mapped Data Cache Example



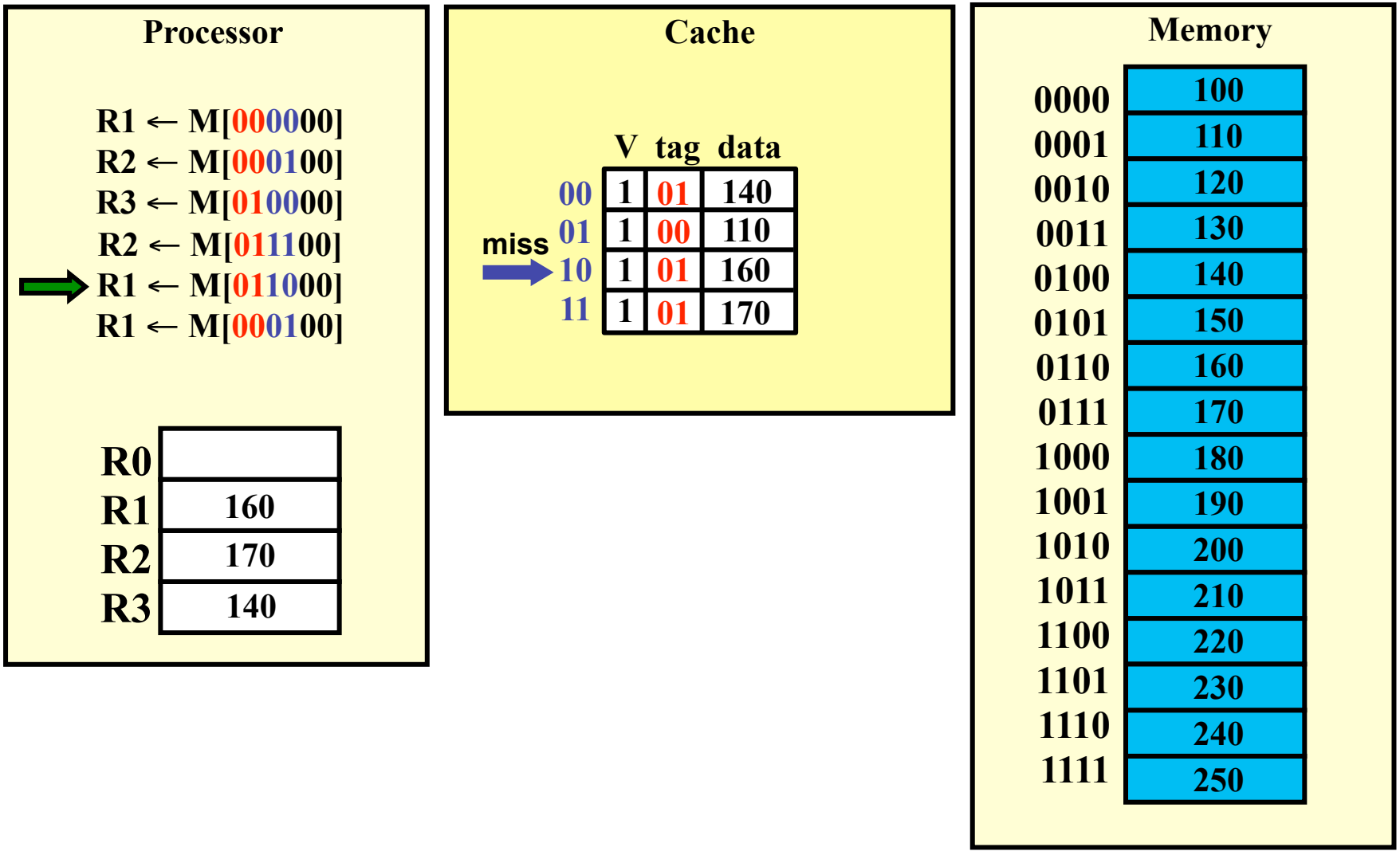
Direct Mapped Data Cache Example



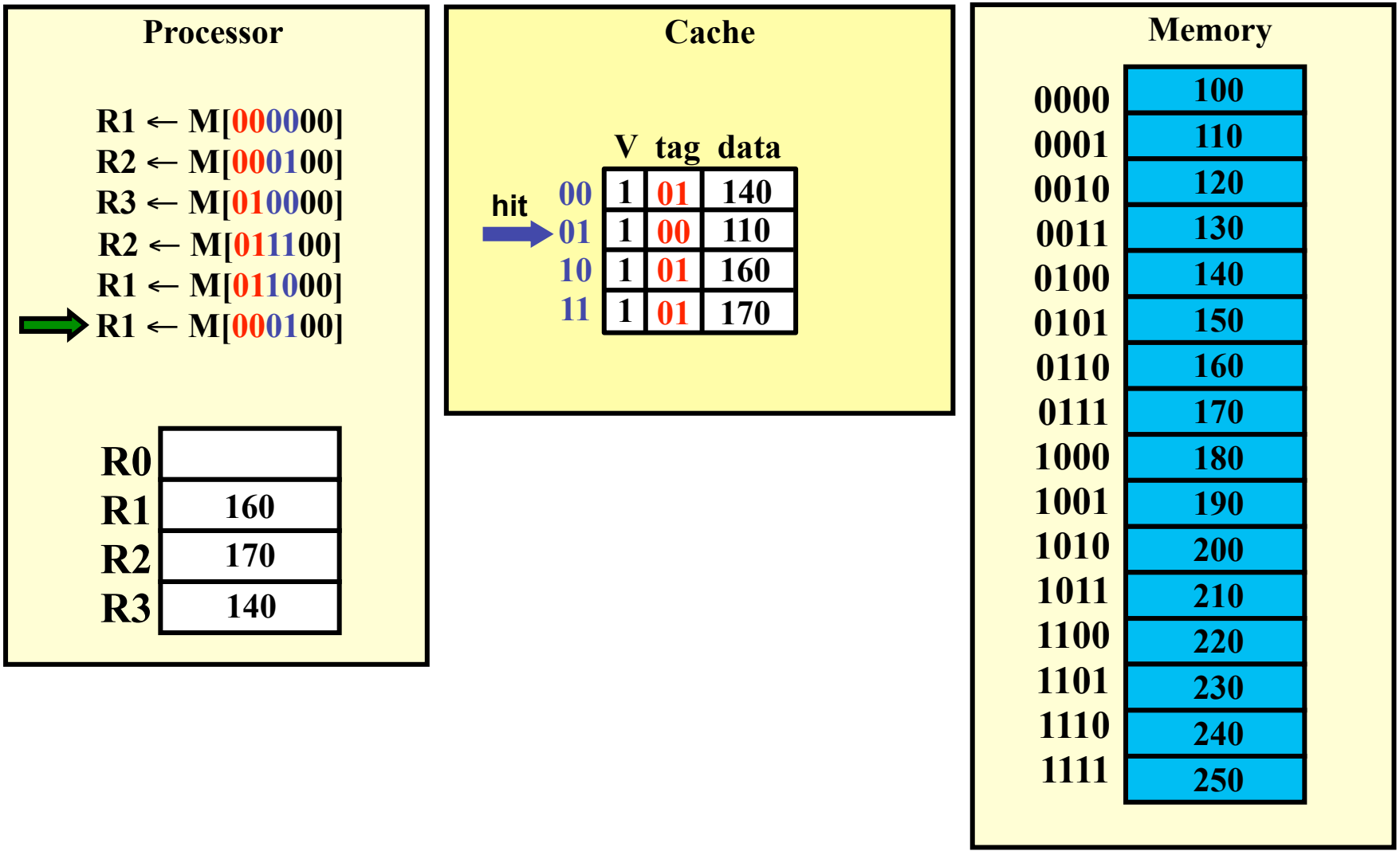
Direct Mapped Data Cache Example



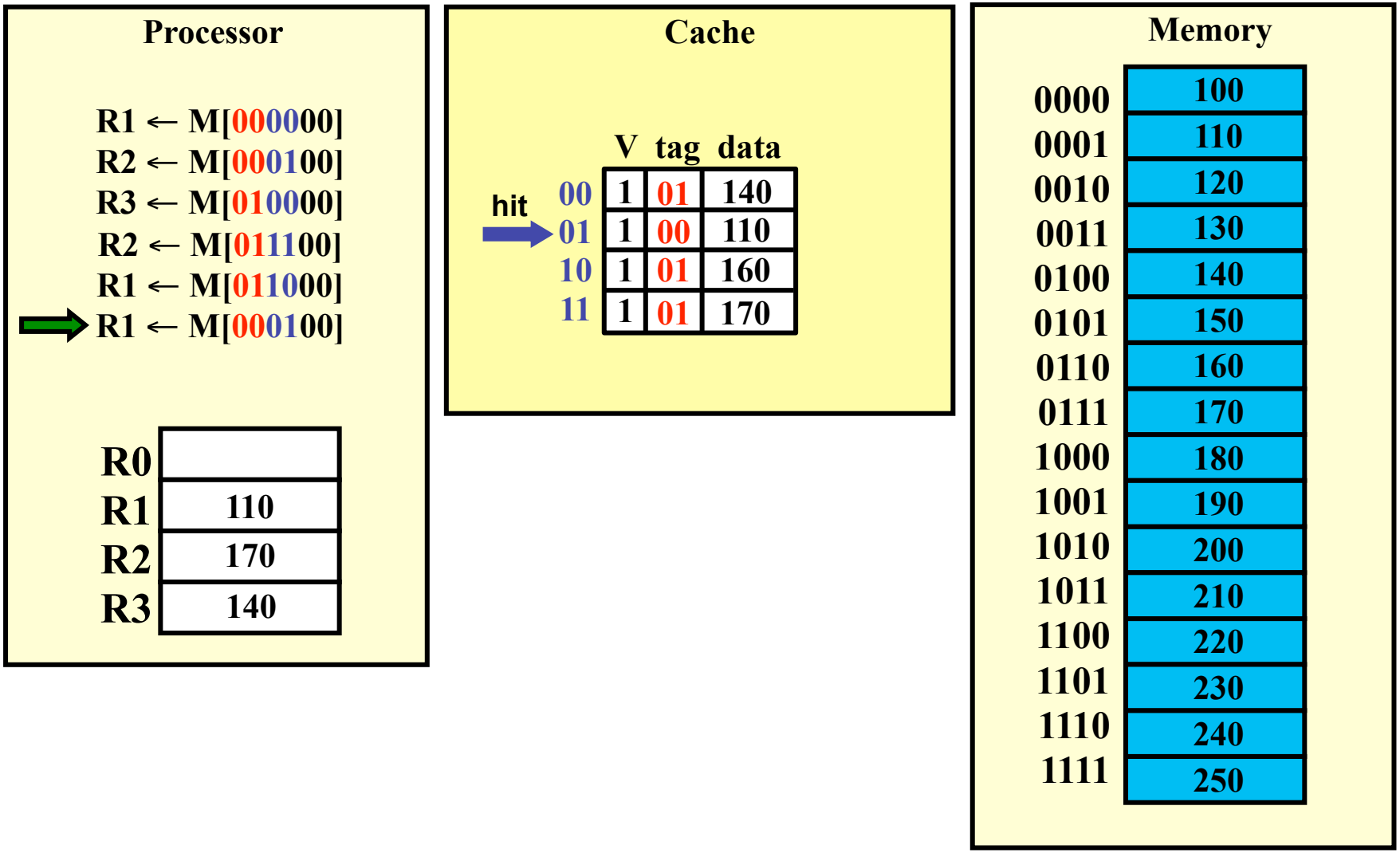
Direct Mapped Data Cache Example



Direct Mapped Data Cache Example

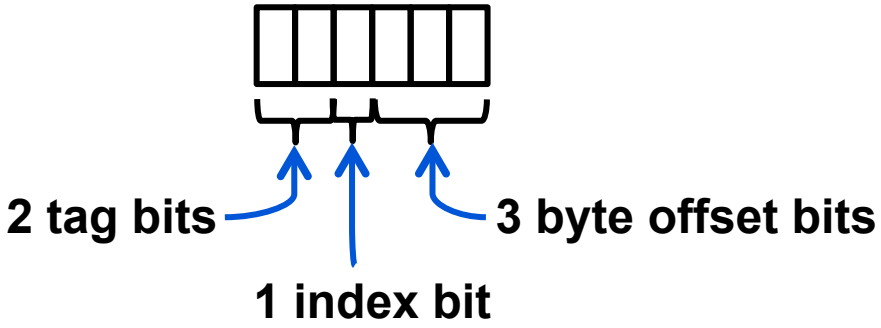


Direct Mapped Data Cache Example

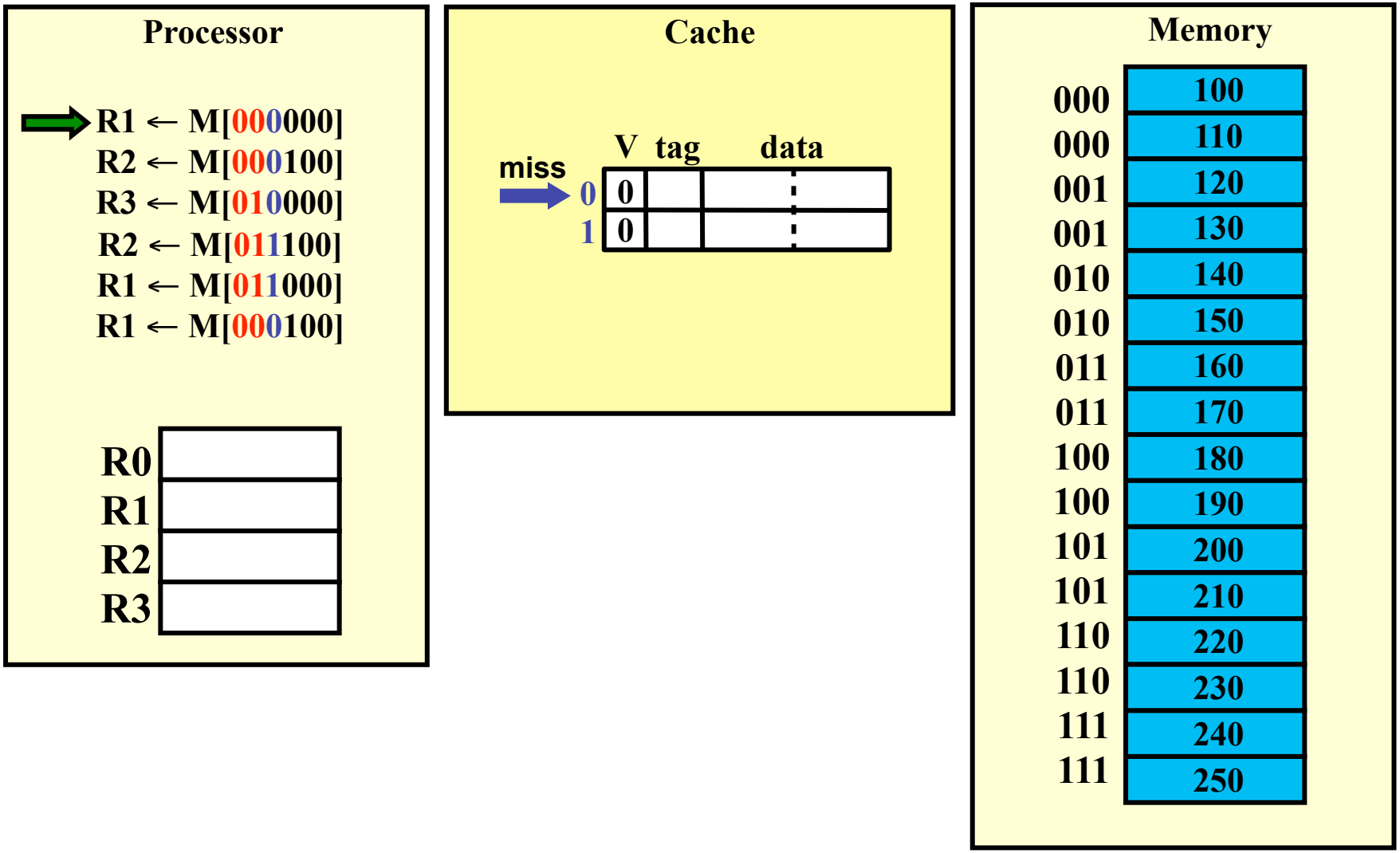


Doubling the Block Size

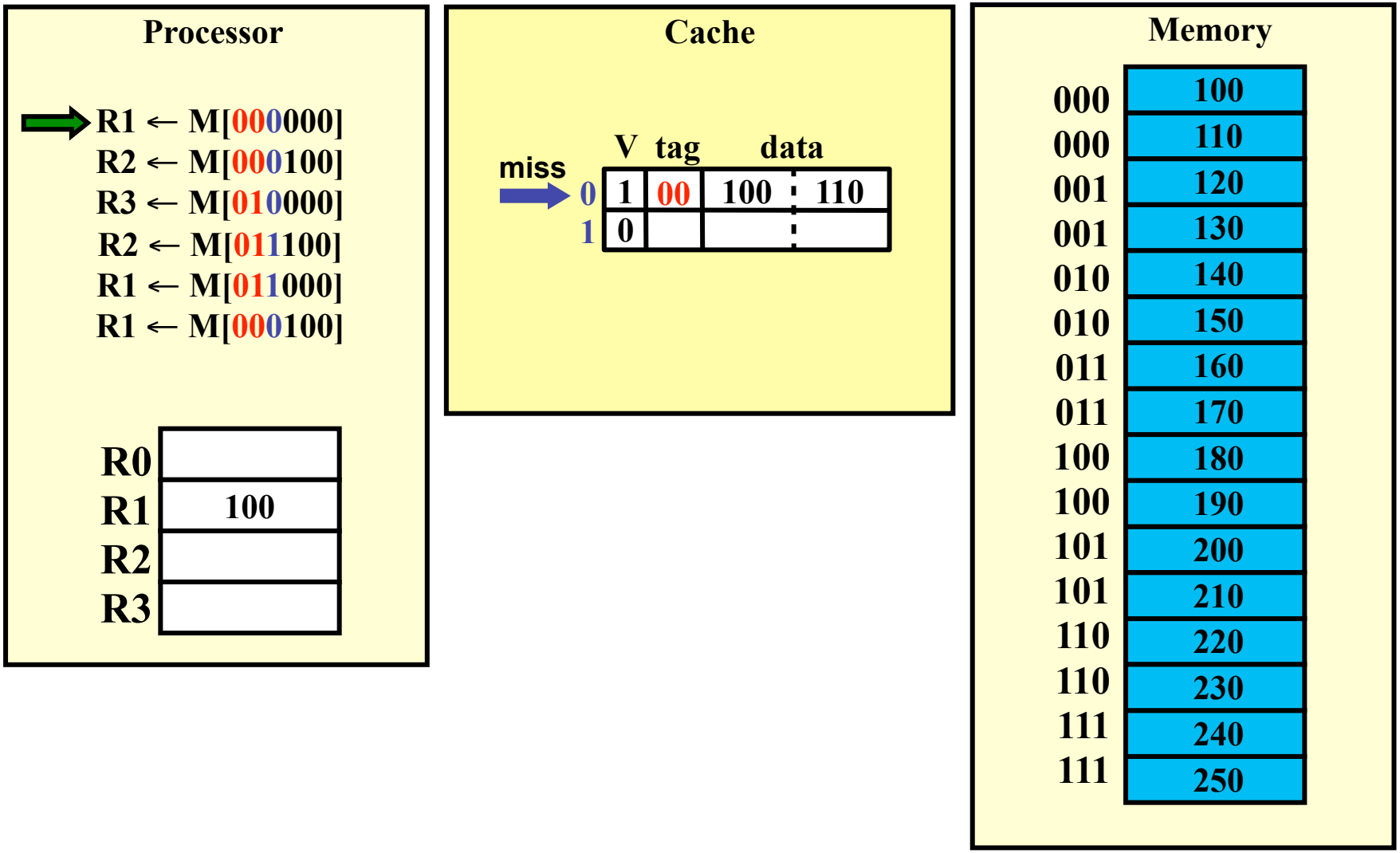
- Size of each block is 8 bytes
- Cache holds 2 blocks
- Memory holds 8 blocks
- Memory address



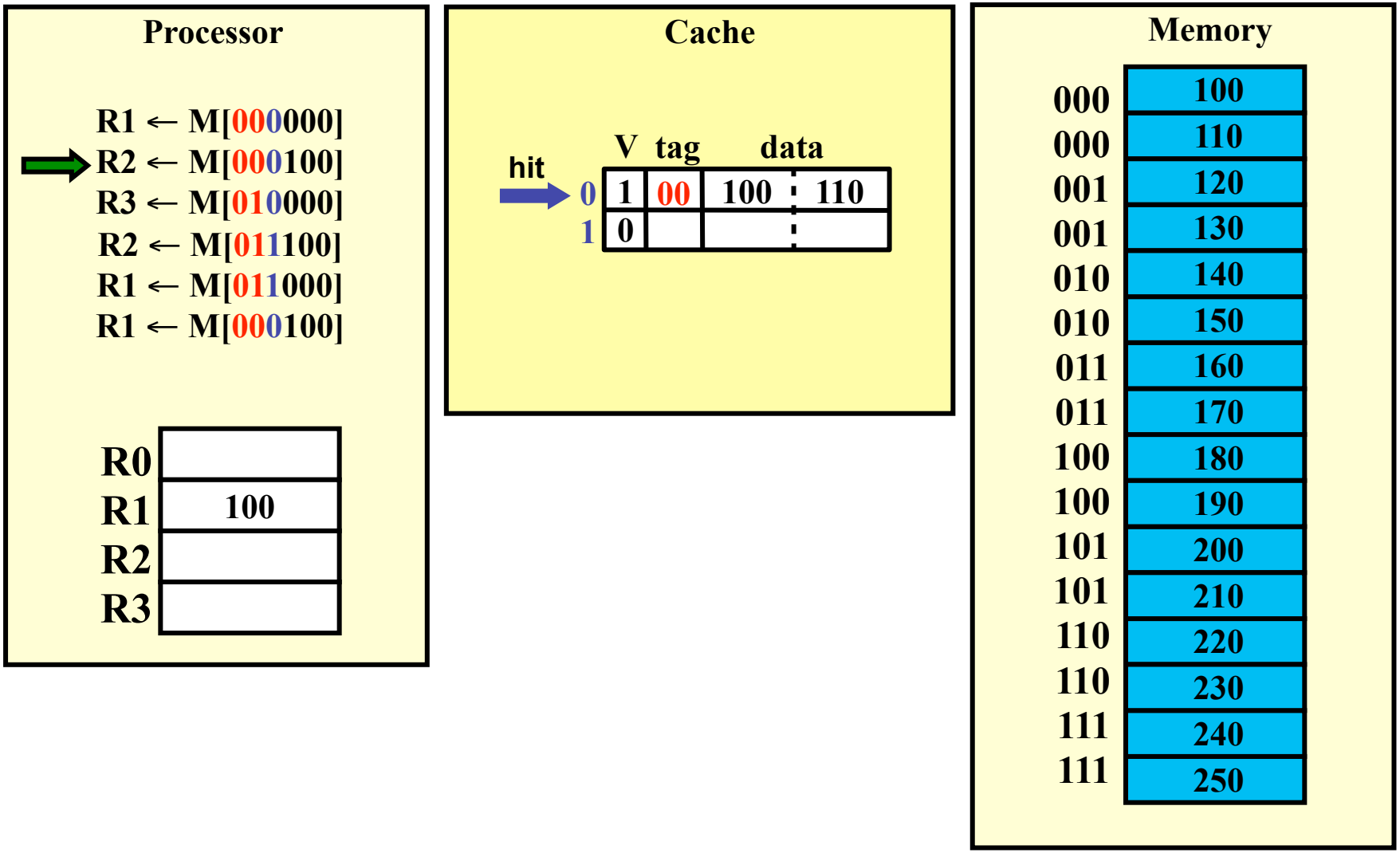
Doubling the Block Size



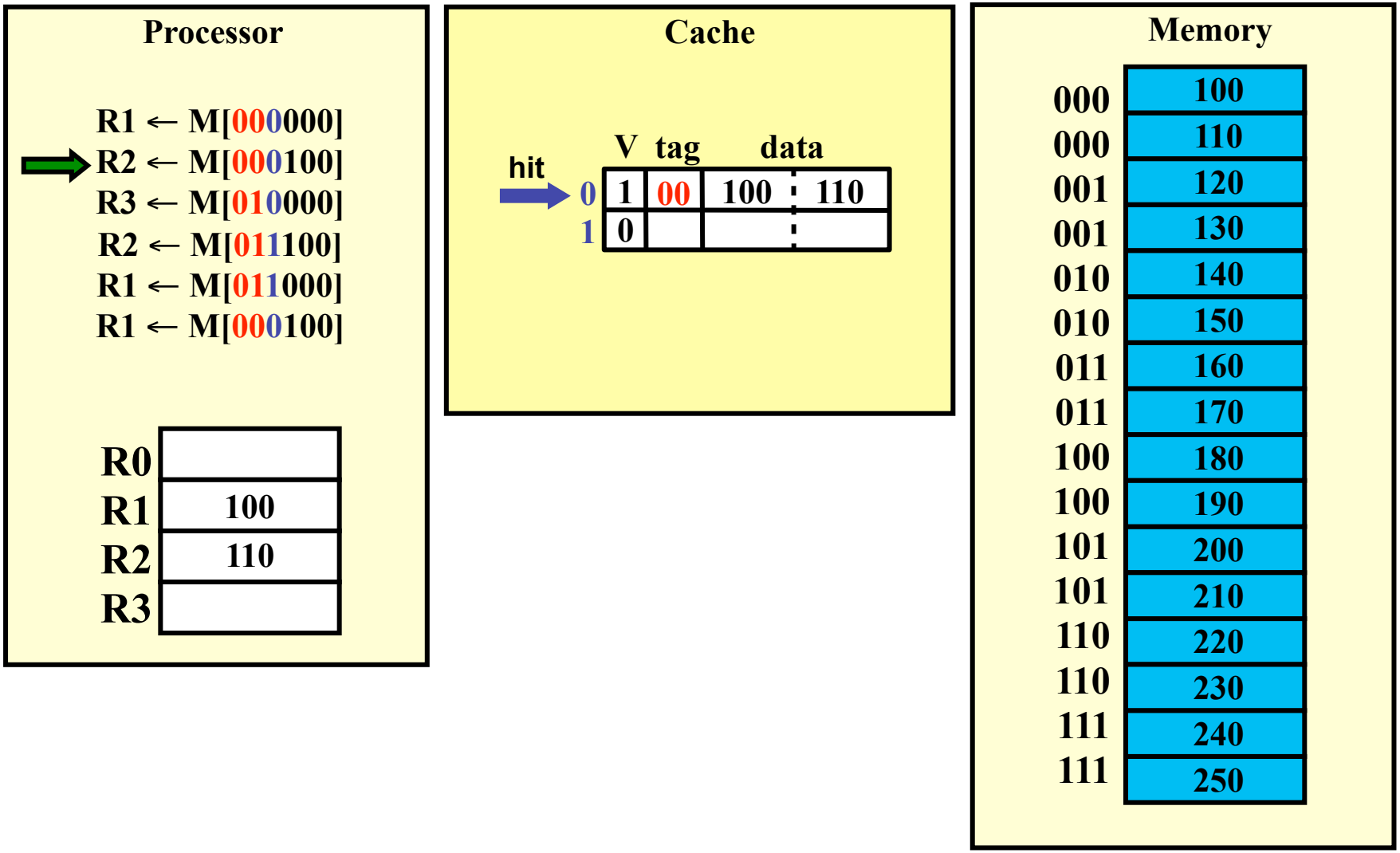
Doubling the Block Size



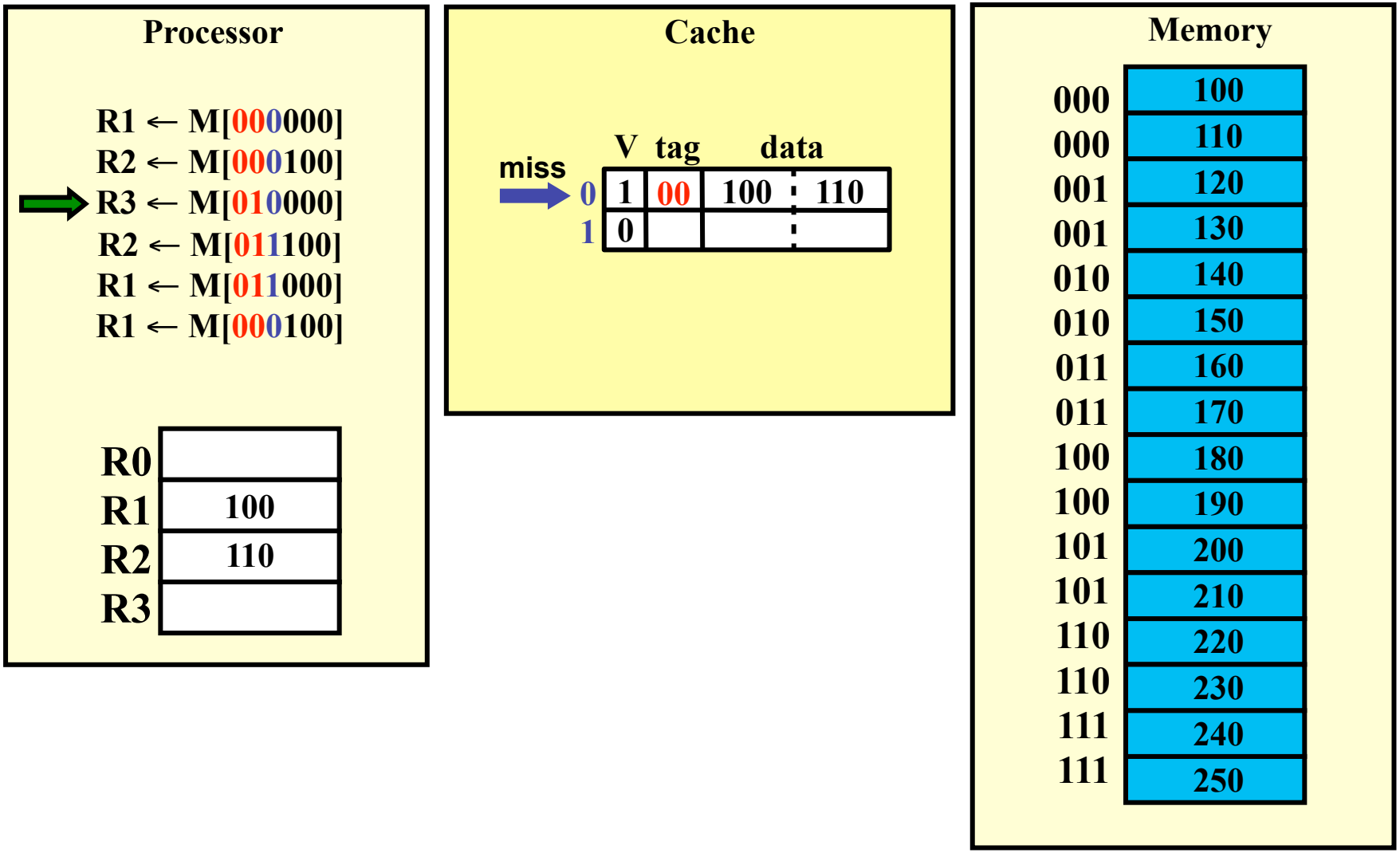
Doubling the Block Size



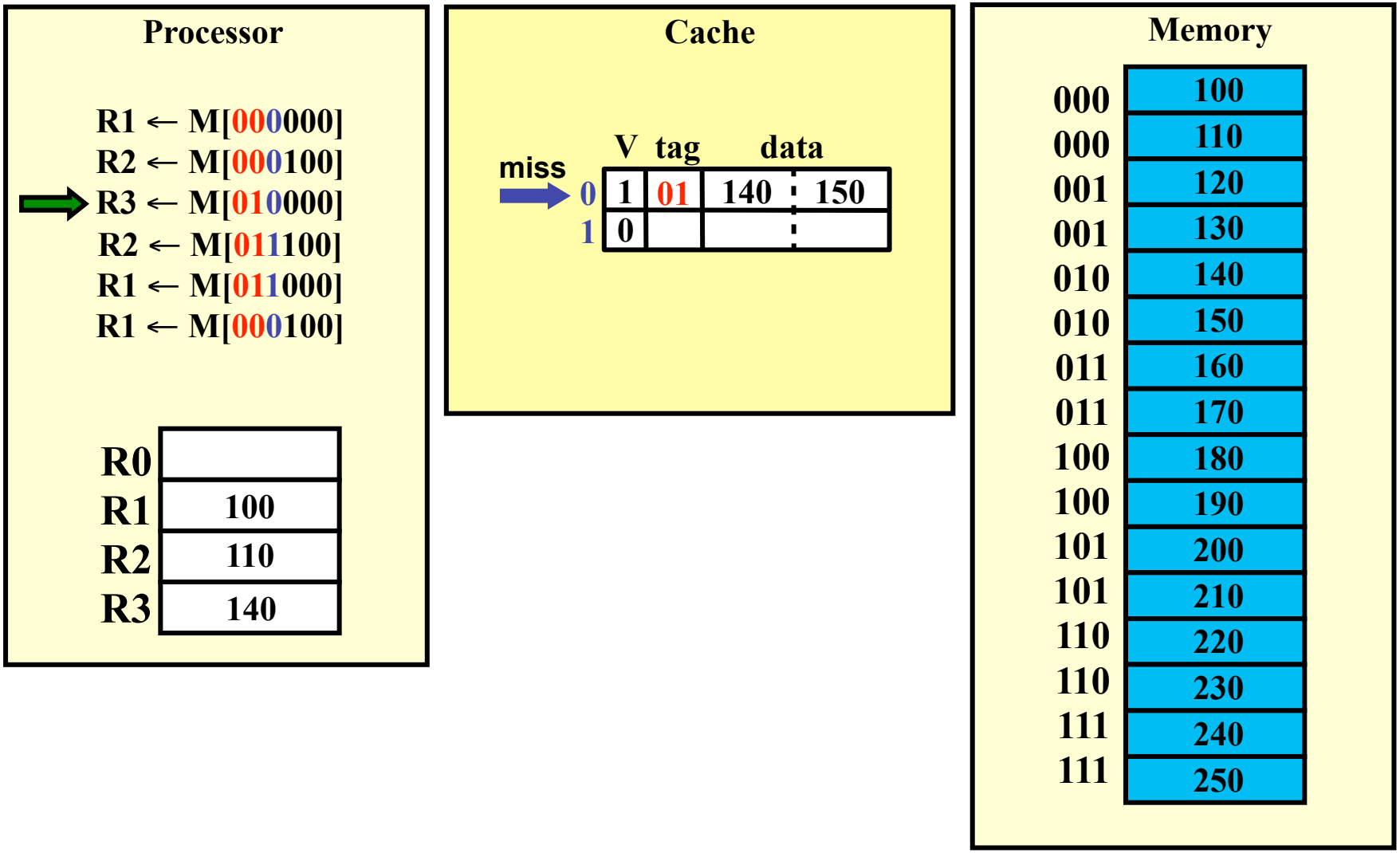
Doubling the Block Size



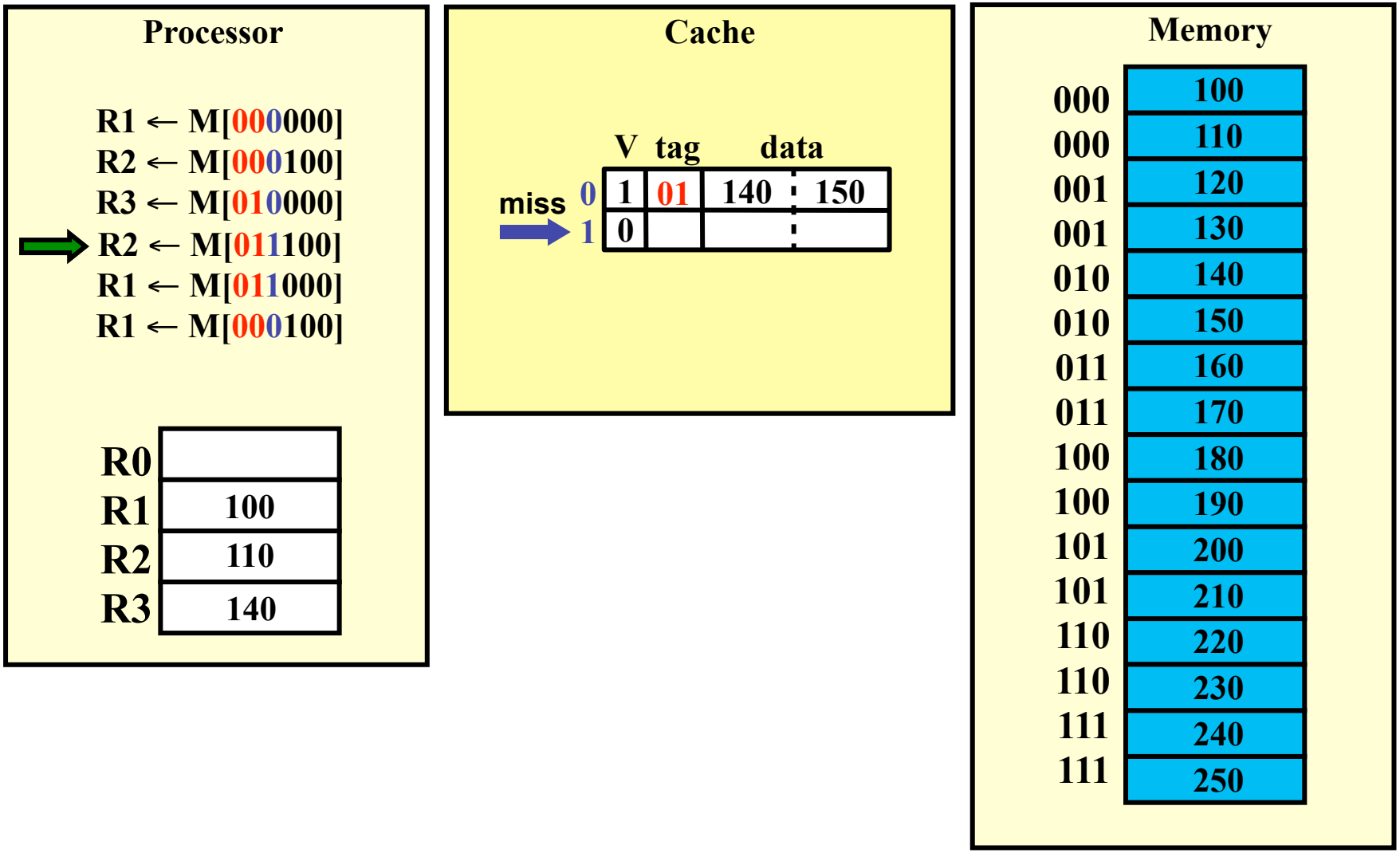
Doubling the Block Size



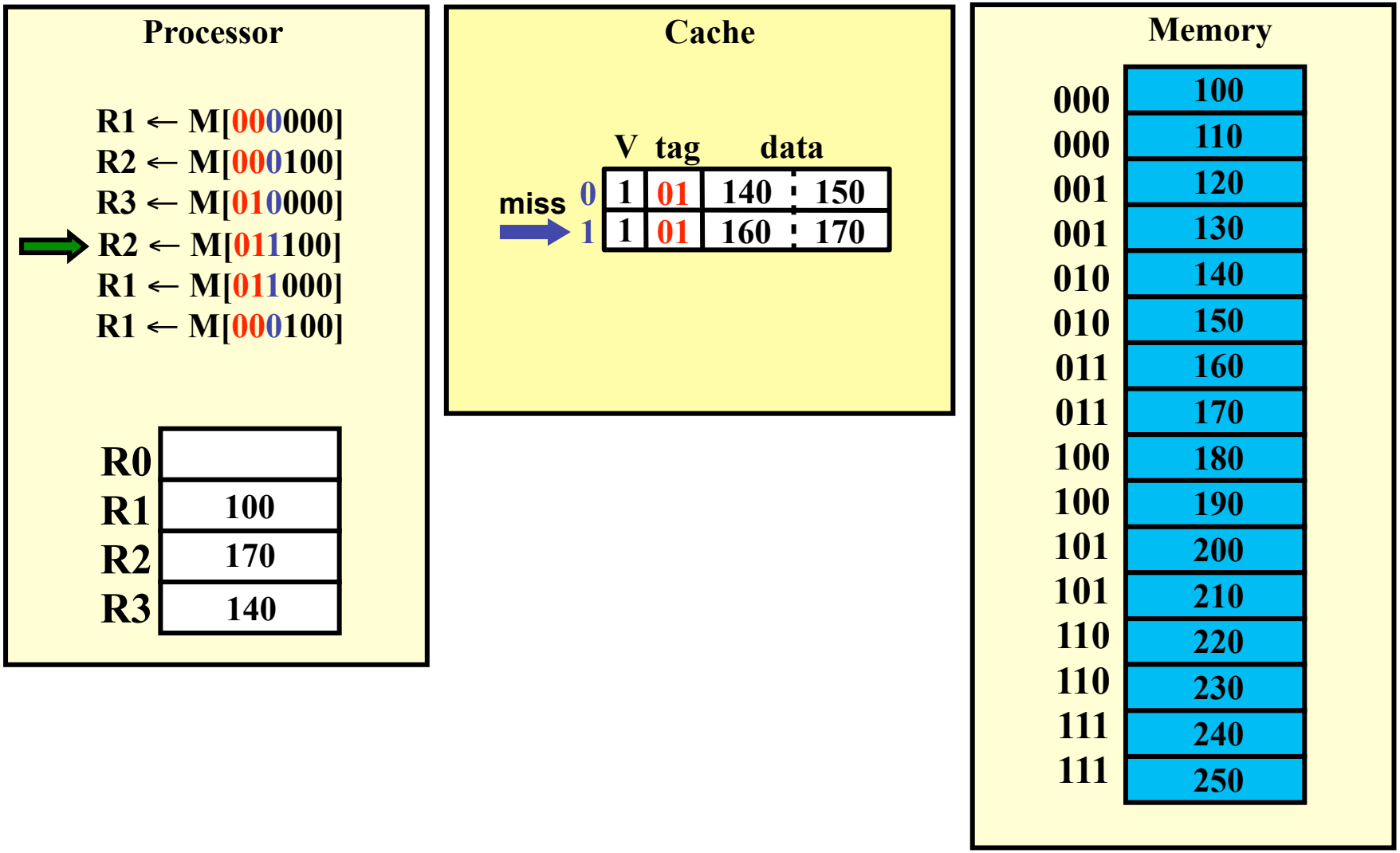
Doubling the Block Size



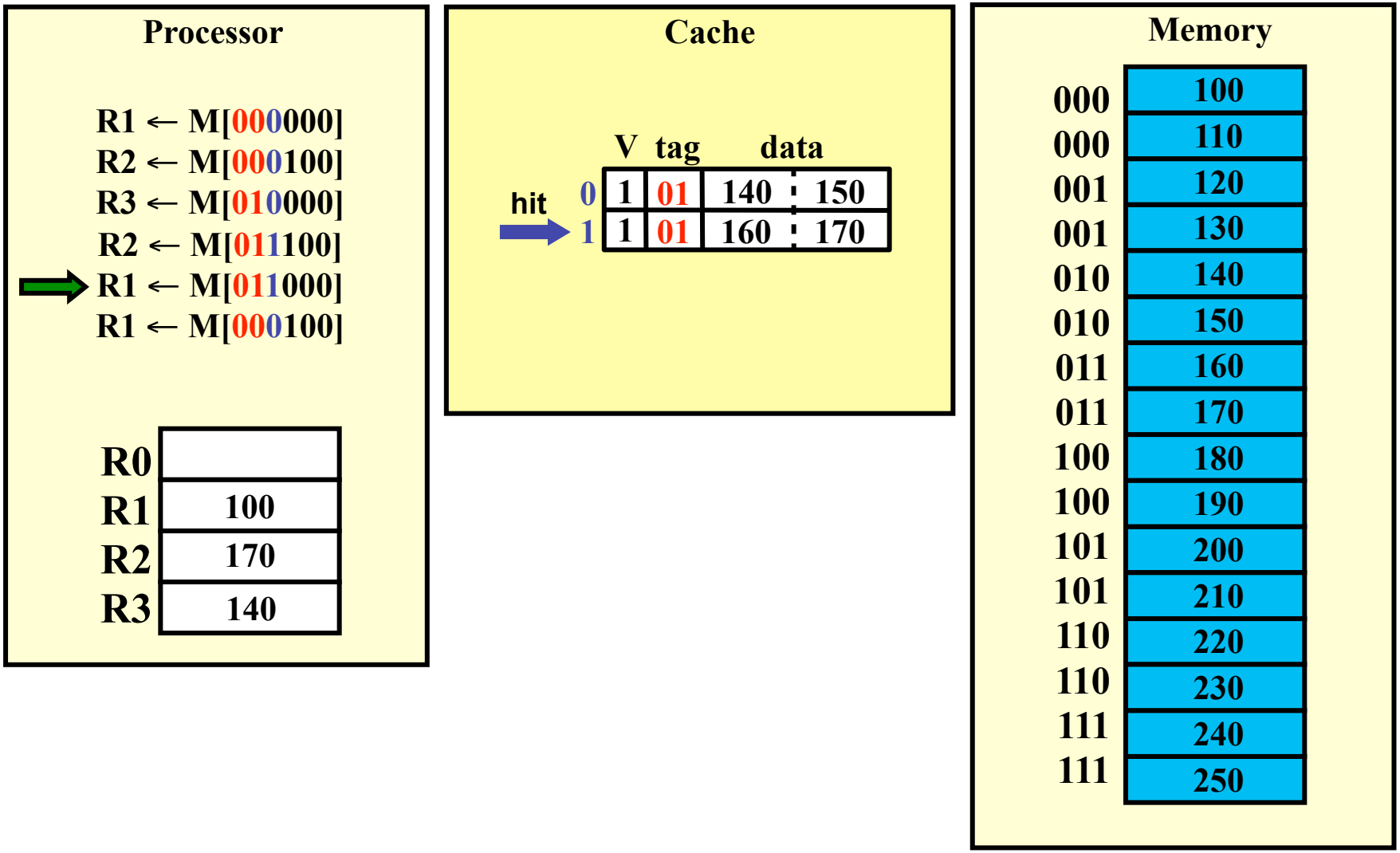
Doubling the Block Size



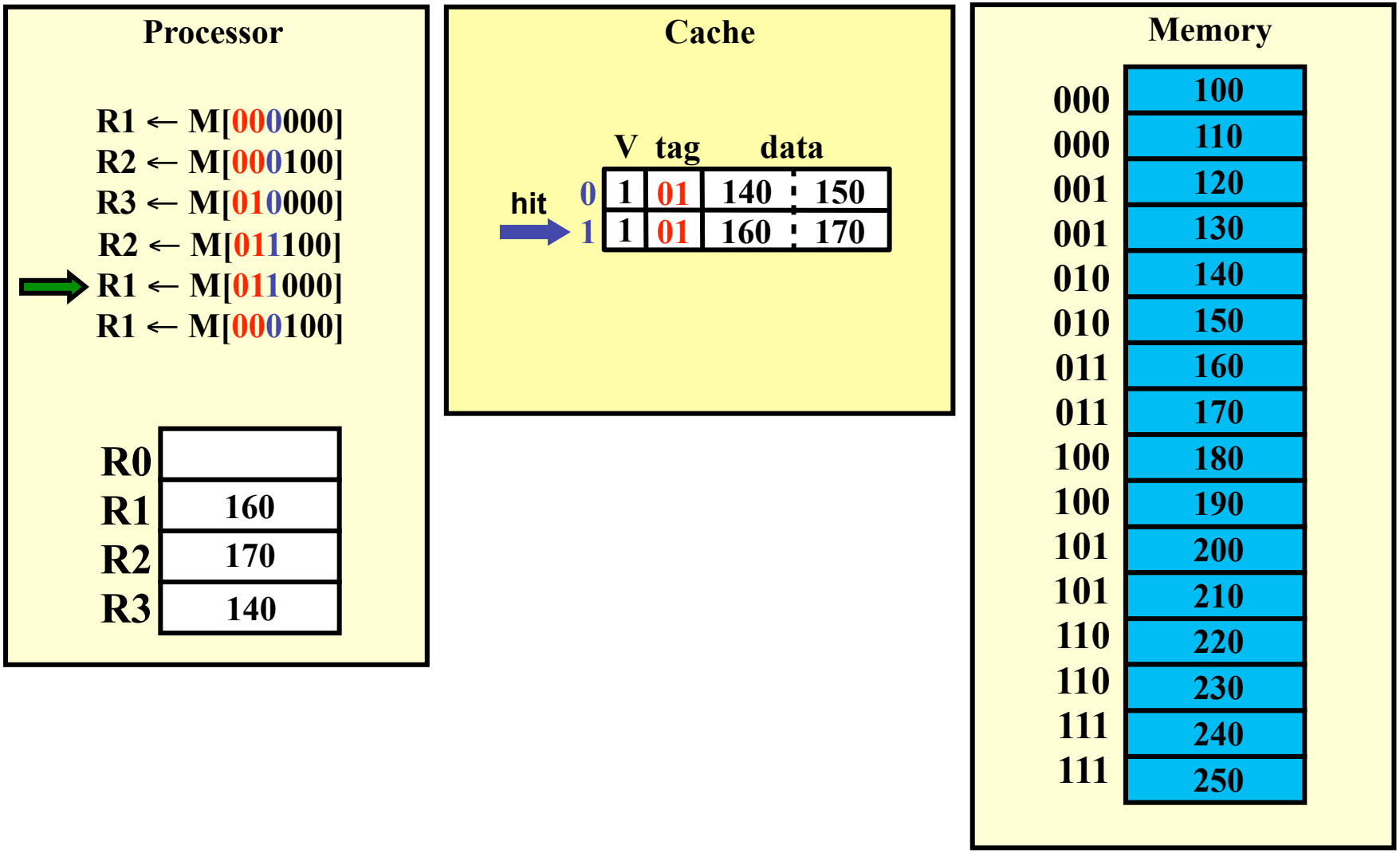
Doubling the Block Size



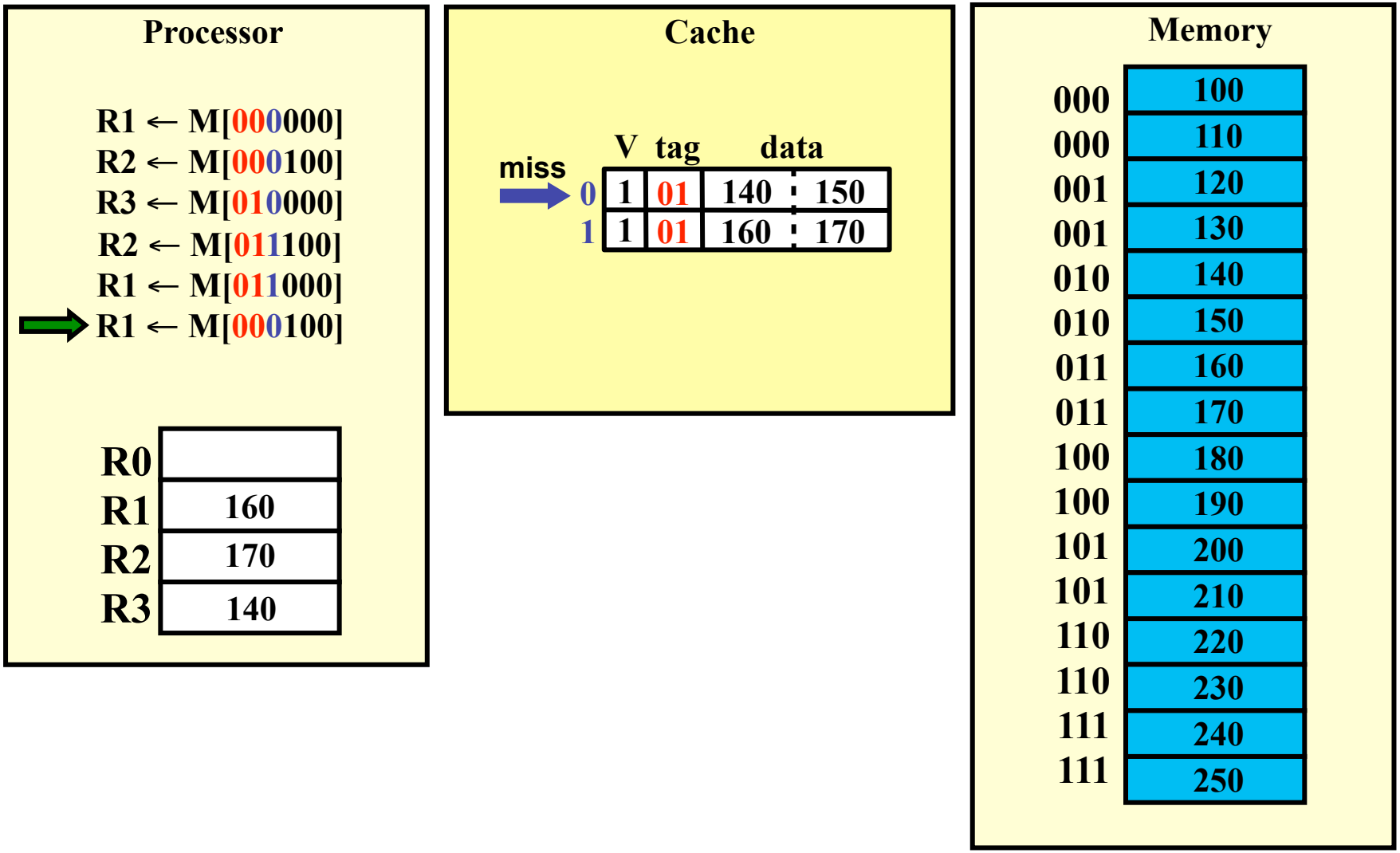
Doubling the Block Size



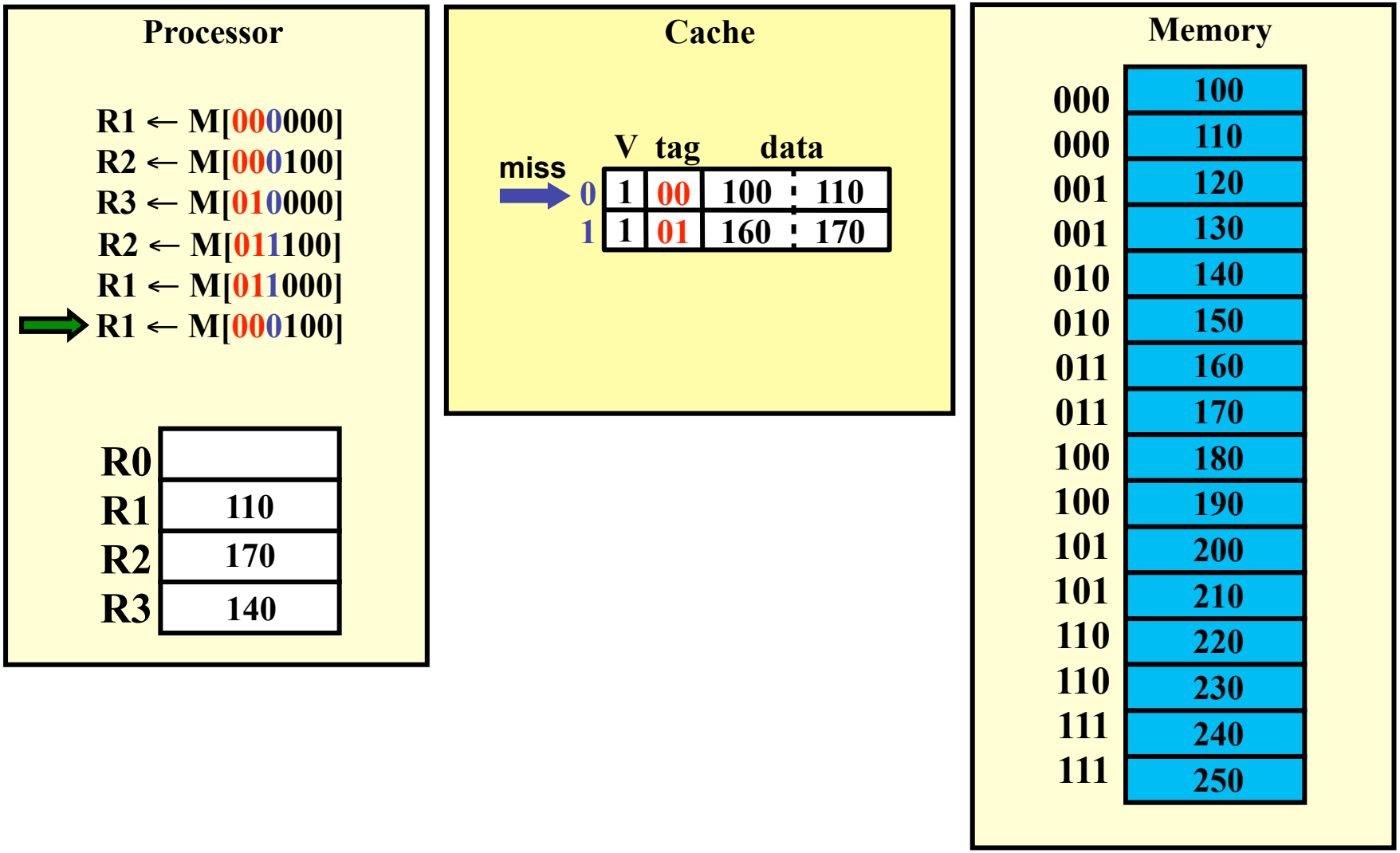
Doubling the Block Size



Doubling the Block Size



Doubling the Block Size



Block Size Tradeoffs

- **Larger blocks may reduce miss rate due to spatial locality**
  - Other data retrieved along with the missed data may be accessed soon
- **But in a fixed-sized cache**
  - Larger blocks ⇒ fewer of them ⇒ increased miss rate due to conflicts
  - Larger blocks ⇒ data fetched along with the requested data may not be used
- **Larger blocks increase the miss penalty**
  - Takes longer to transfer a larger block from memory

Associative Caches

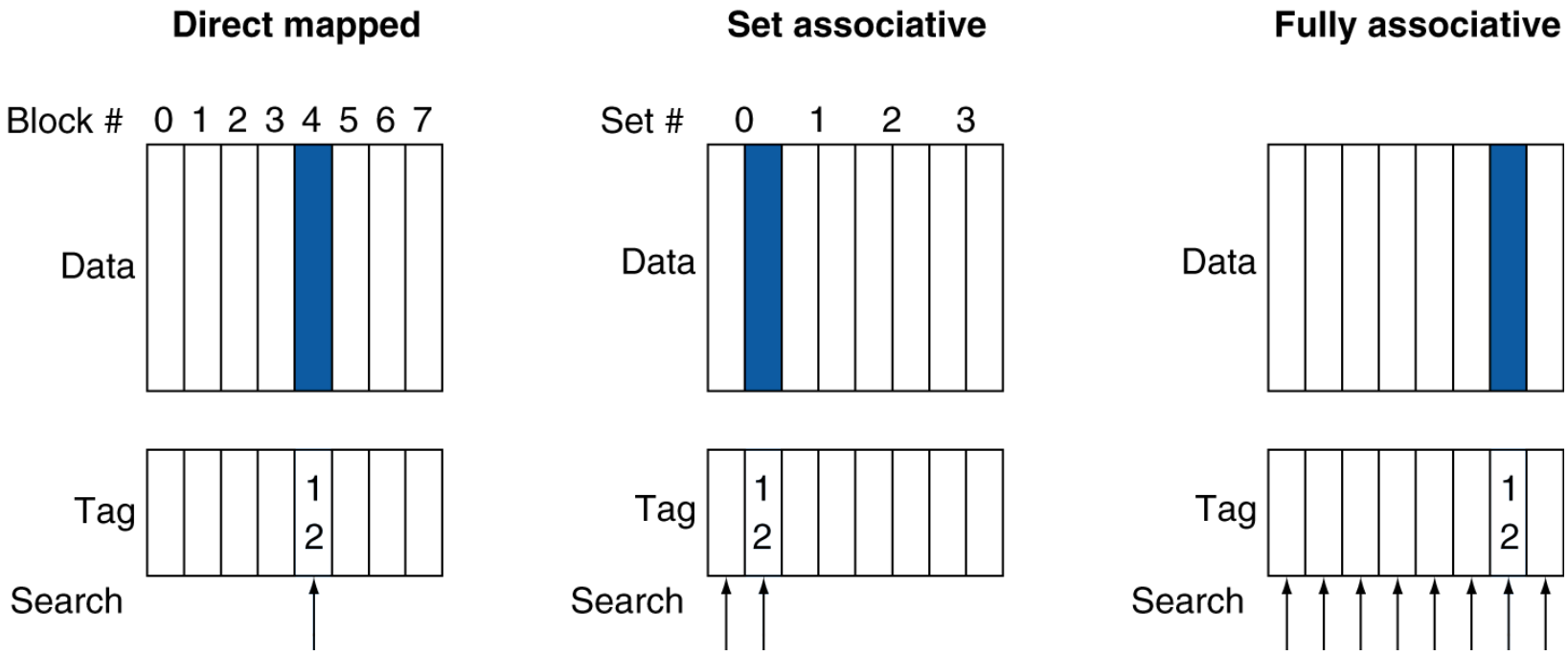
- Provide more flexible placement of blocks
- Each *set* in the cache contains >1 block
  - A block can be stored anywhere in the set
  - A set is searched in parallel to find a block
- Increasing the associatively...
  - Decreases the miss rate (fewer conflicts)
  - Increases the hit time (takes longer to search)

Associative Caches

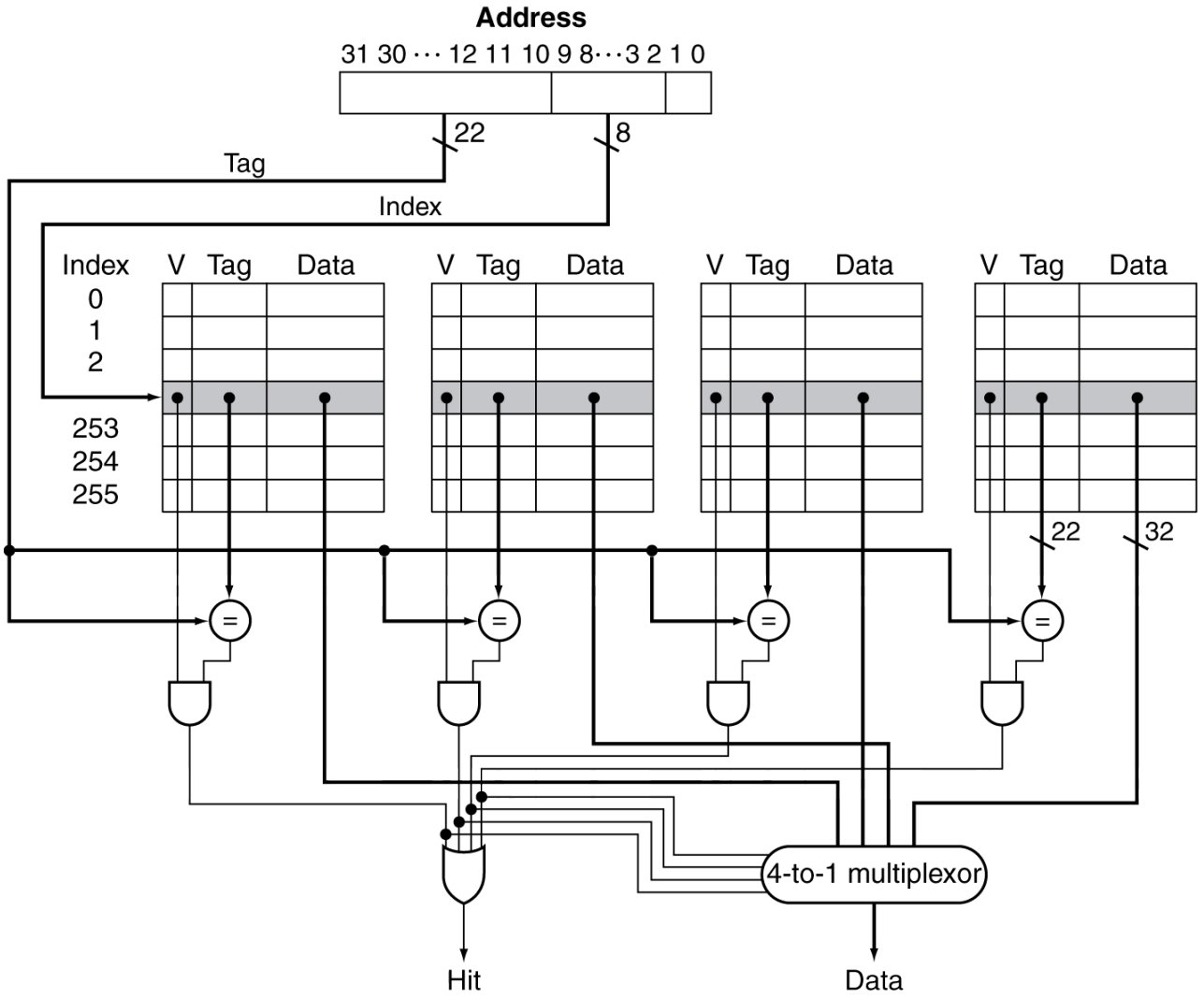
- **Fully associative**
  - Block can go in any cache location
  - All entries are searched at once
  - Comparator per entry (expensive)

- ***n*-way set associative**
  - Each set contains *n* entries (*ways*)
  - Block number determines which set
  - All ways in the selected set are searched at once
  - *n* comparators (less expensive)

Associative Cache Examples

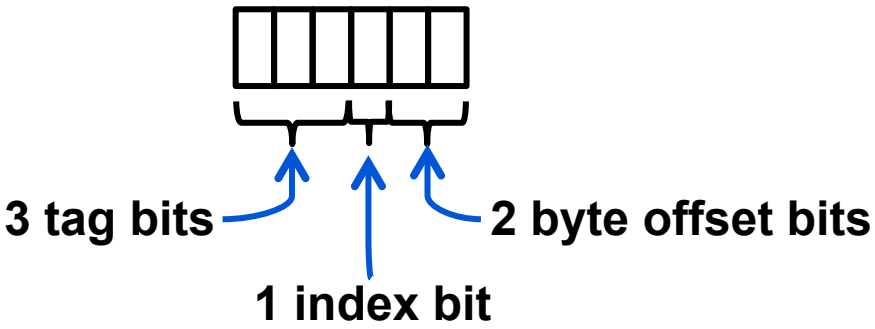


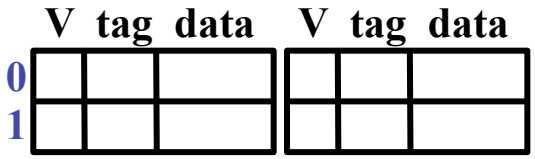
4-way Set Associative Cache



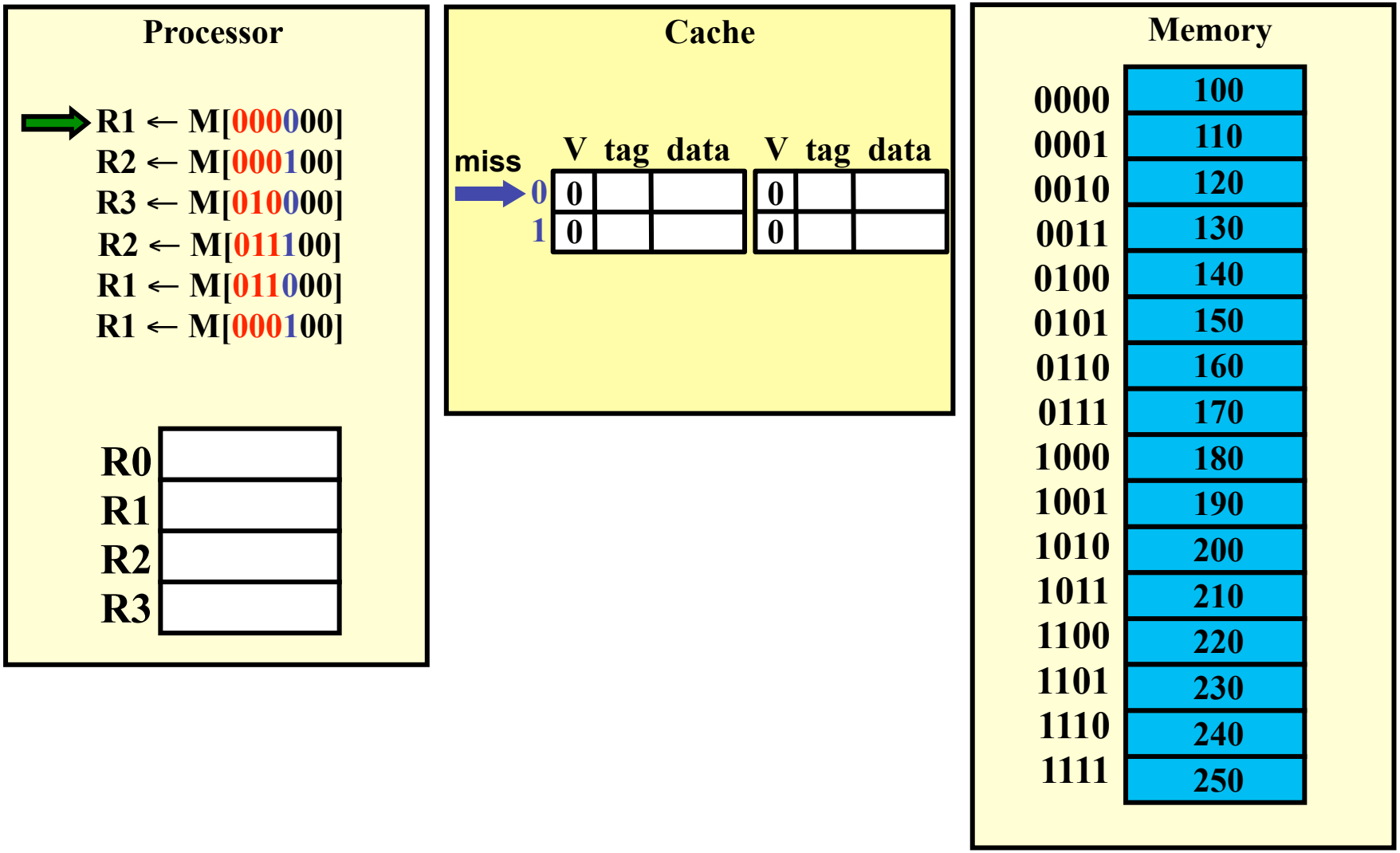
2-way Set Associative Example

- Size of each block is 4 bytes
- Cache holds 4 blocks, 2-way set associative
- Memory holds 16 blocks
- Memory address

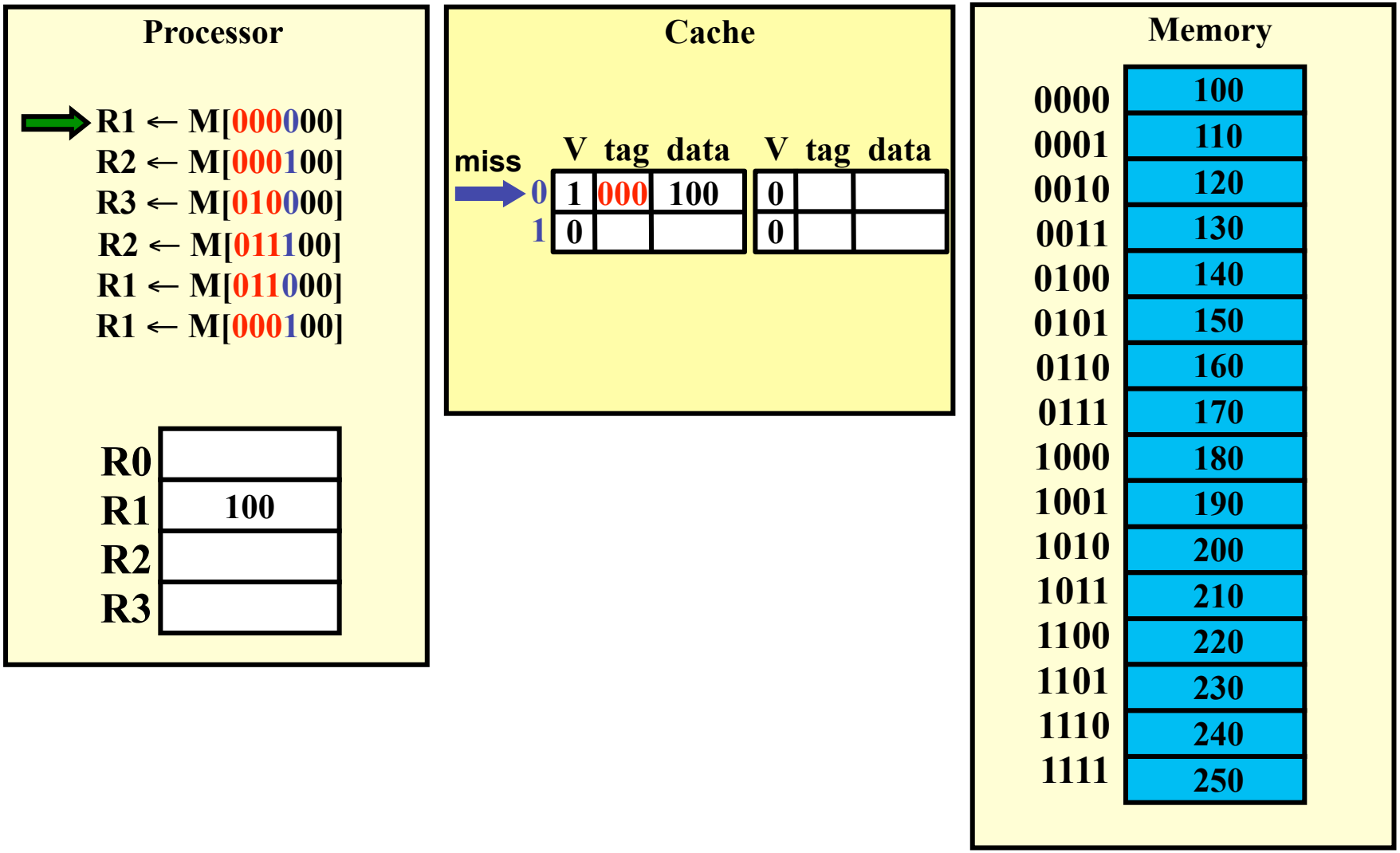




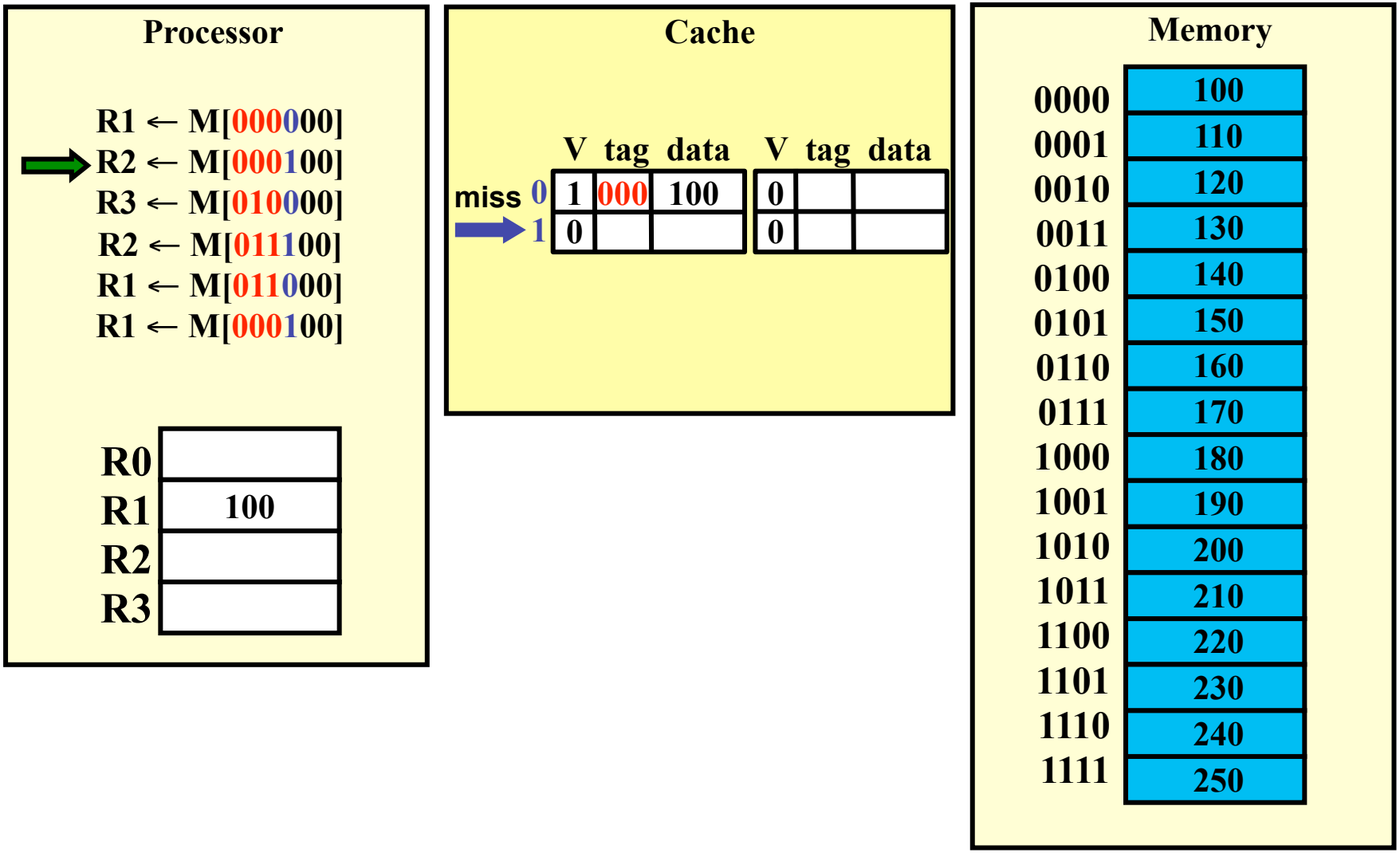
2-way Set Associative Example



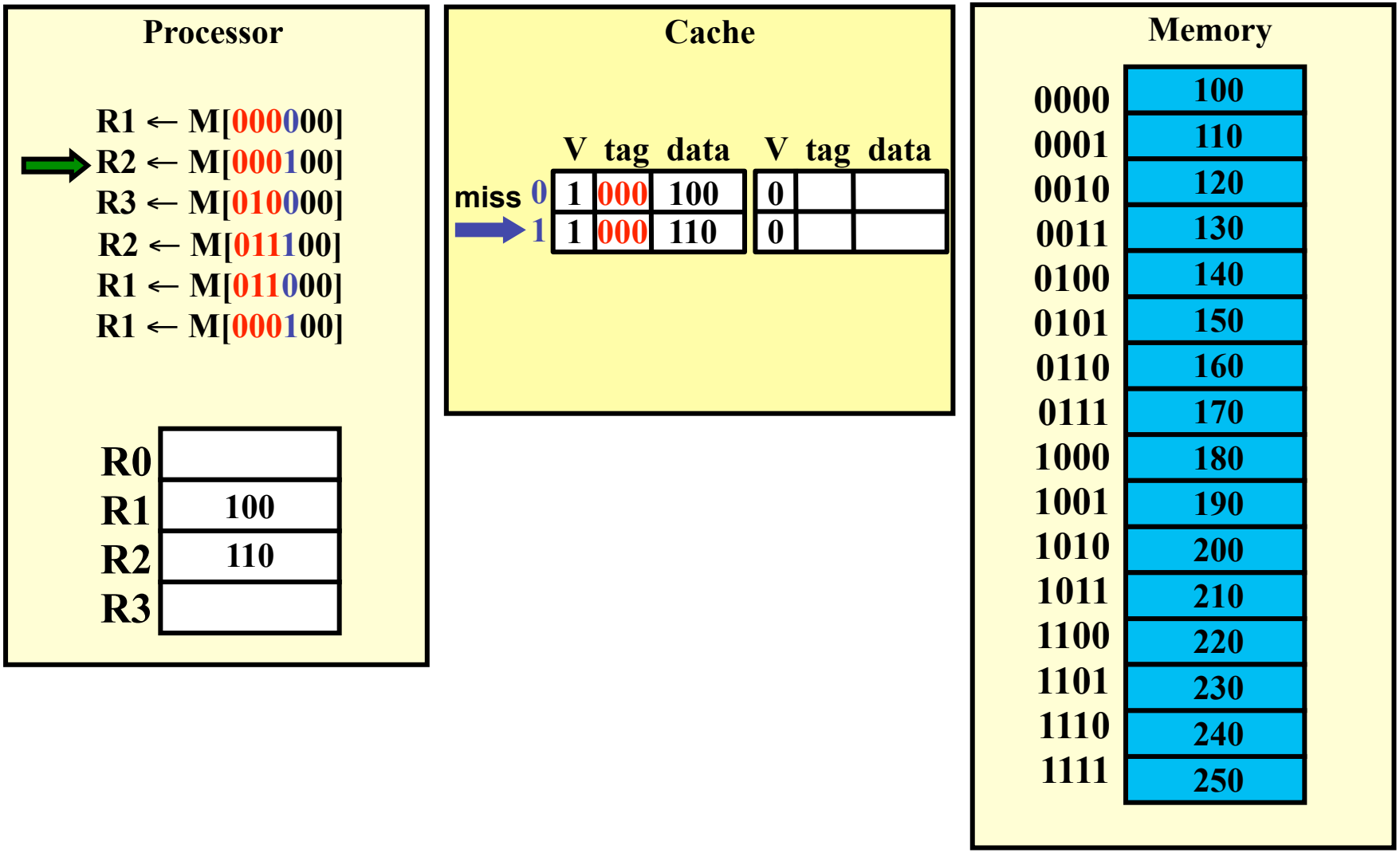
2-way Set Associative Example



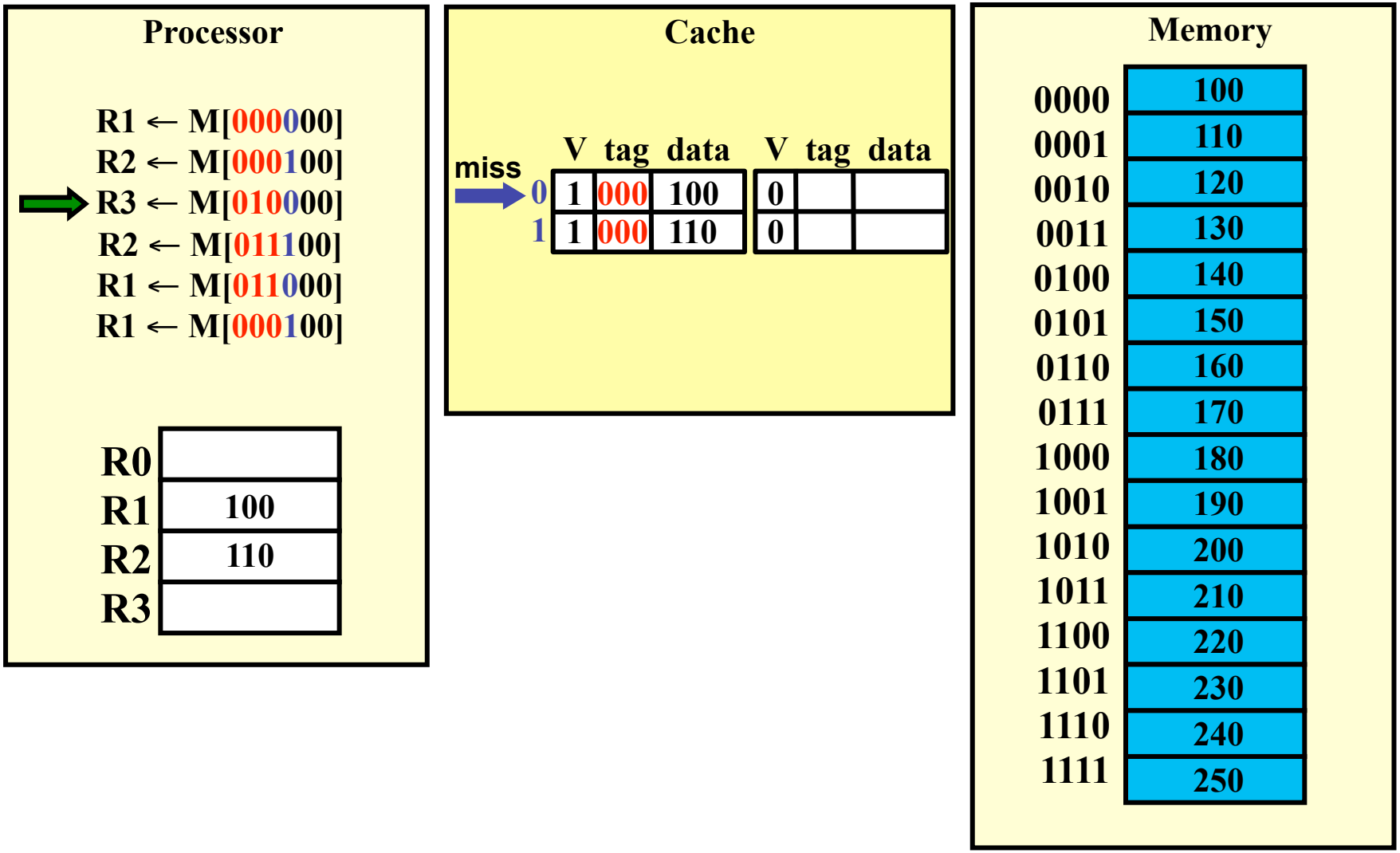
2-way Set Associative Example



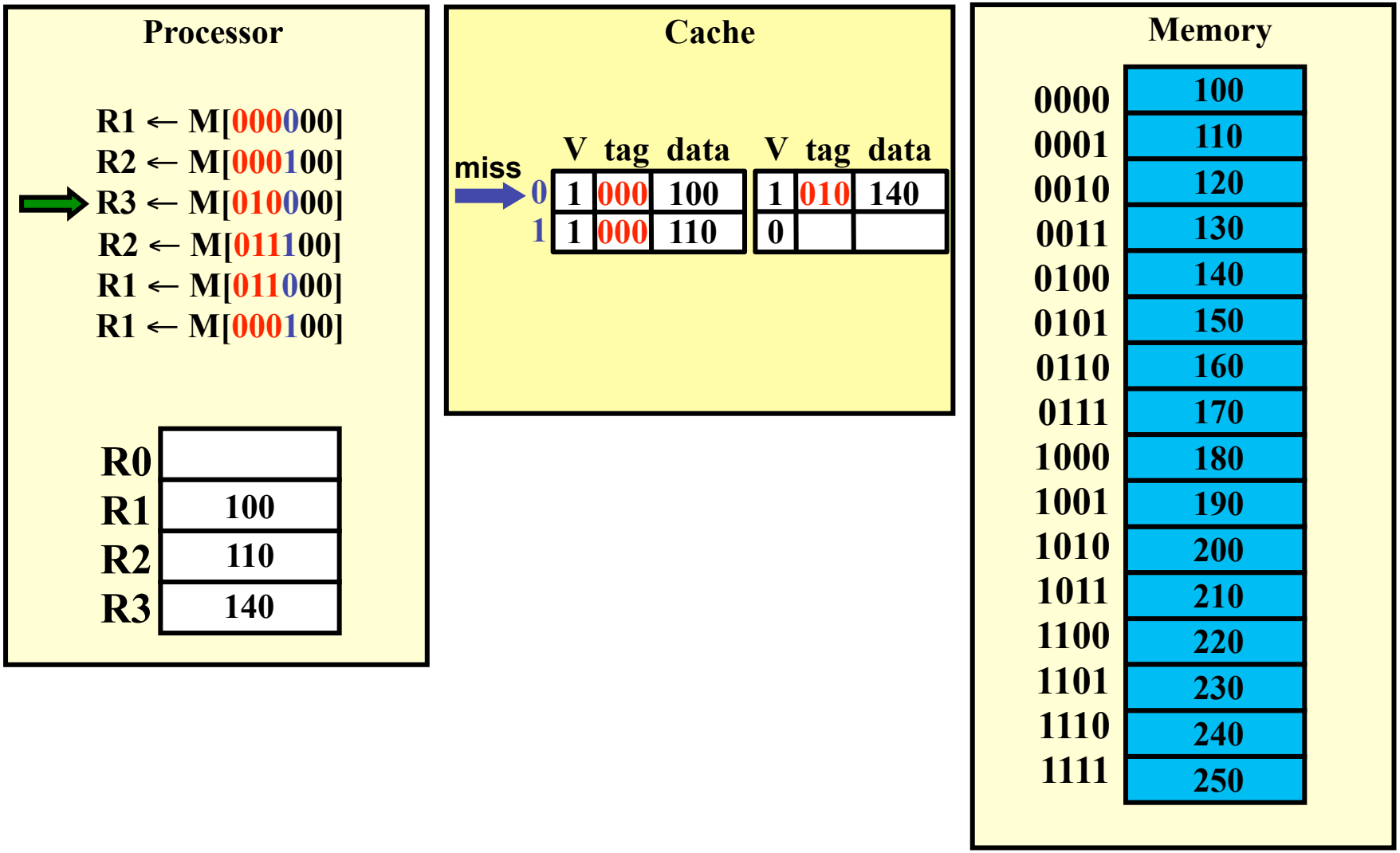
2-way Set Associative Example



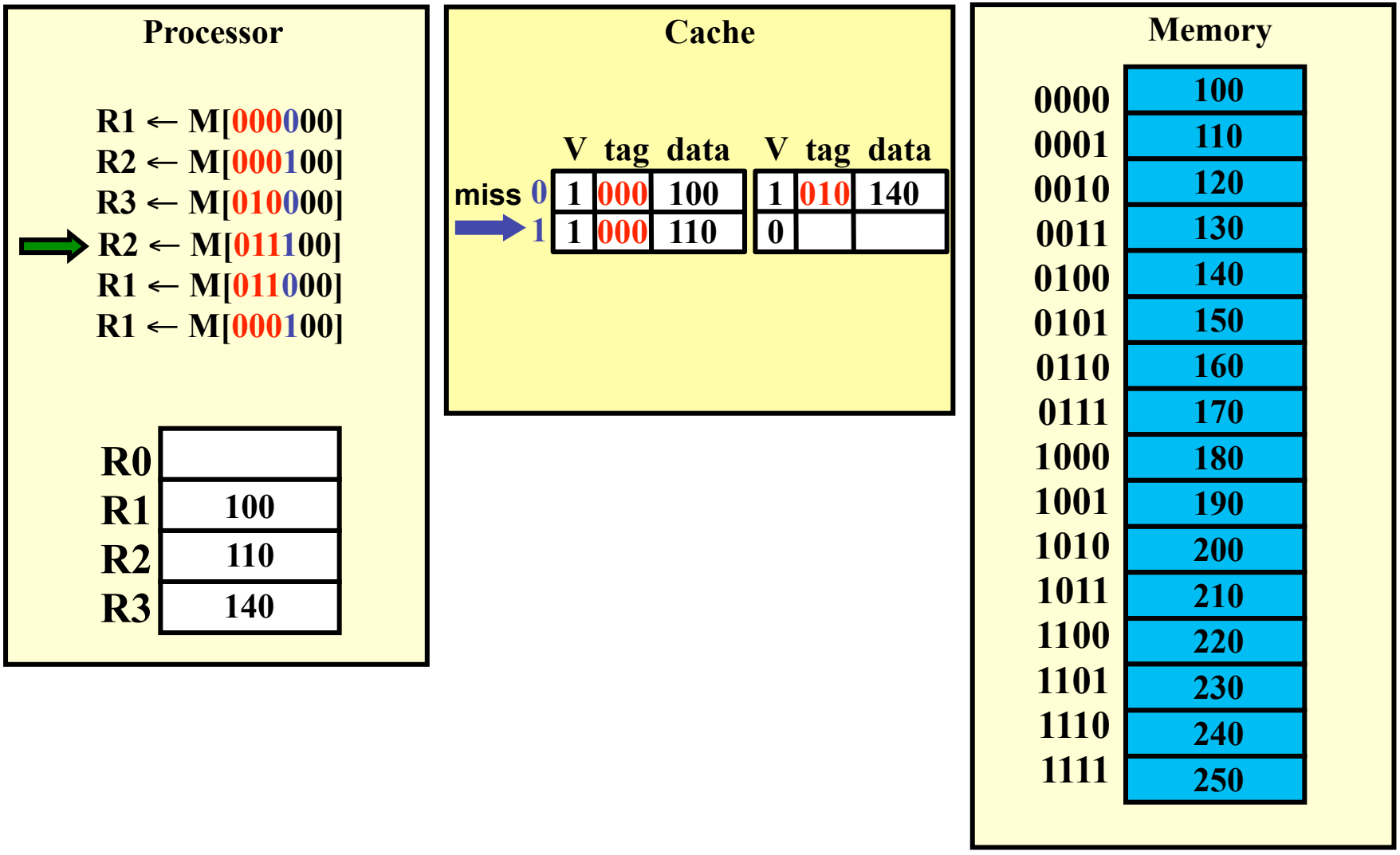
2-way Set Associative Example



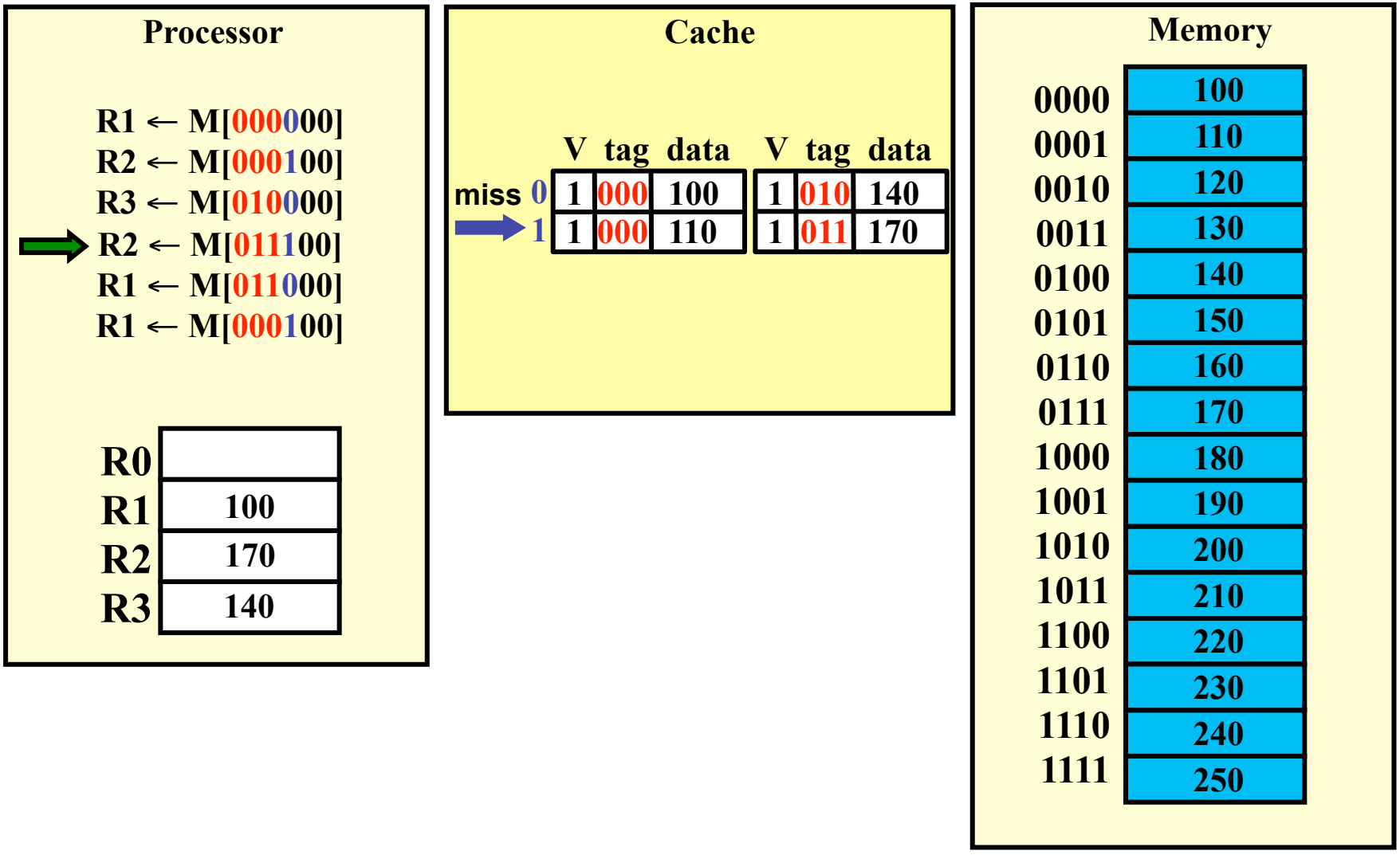
2-way Set Associative Example



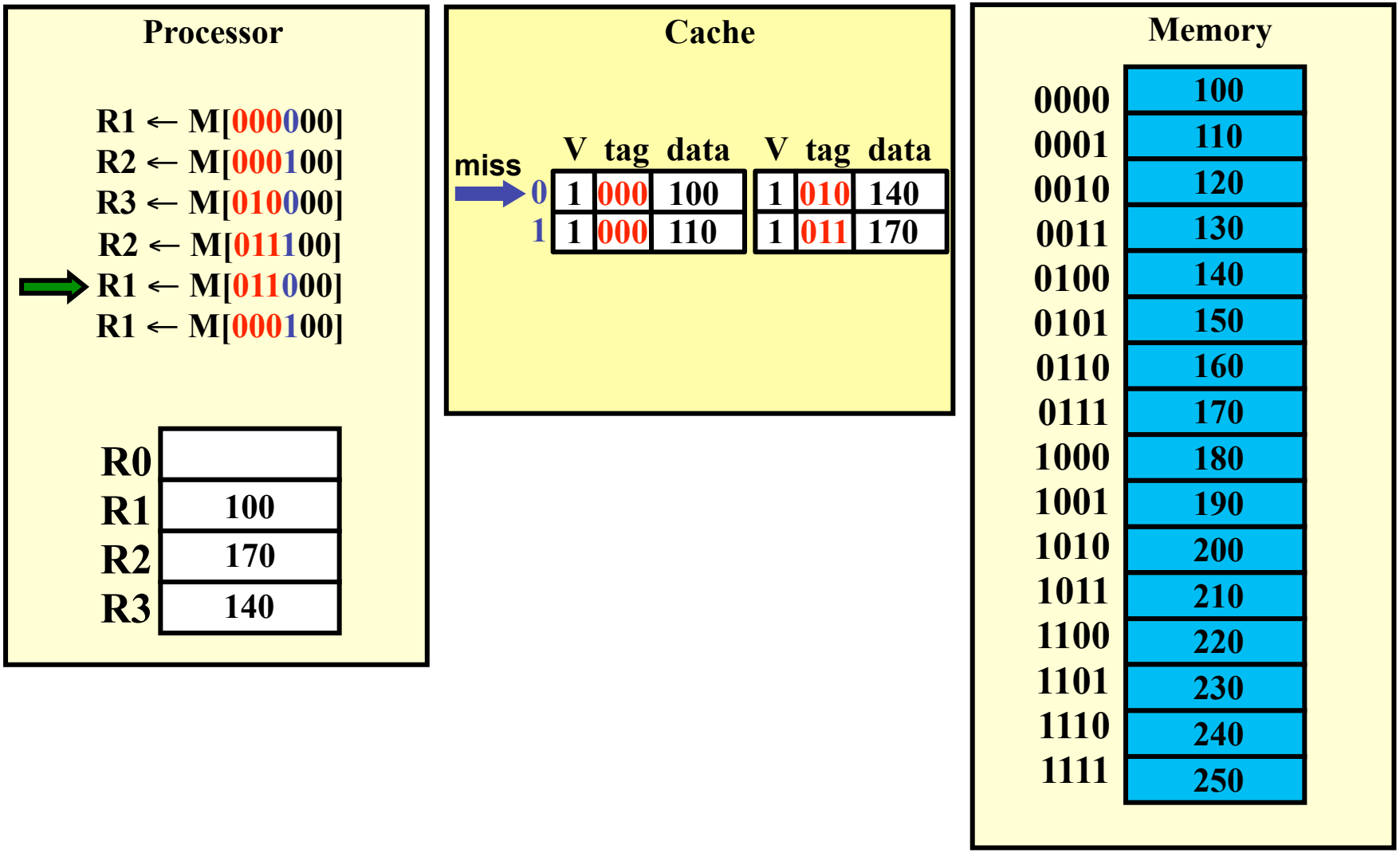
2-way Set Associative Example



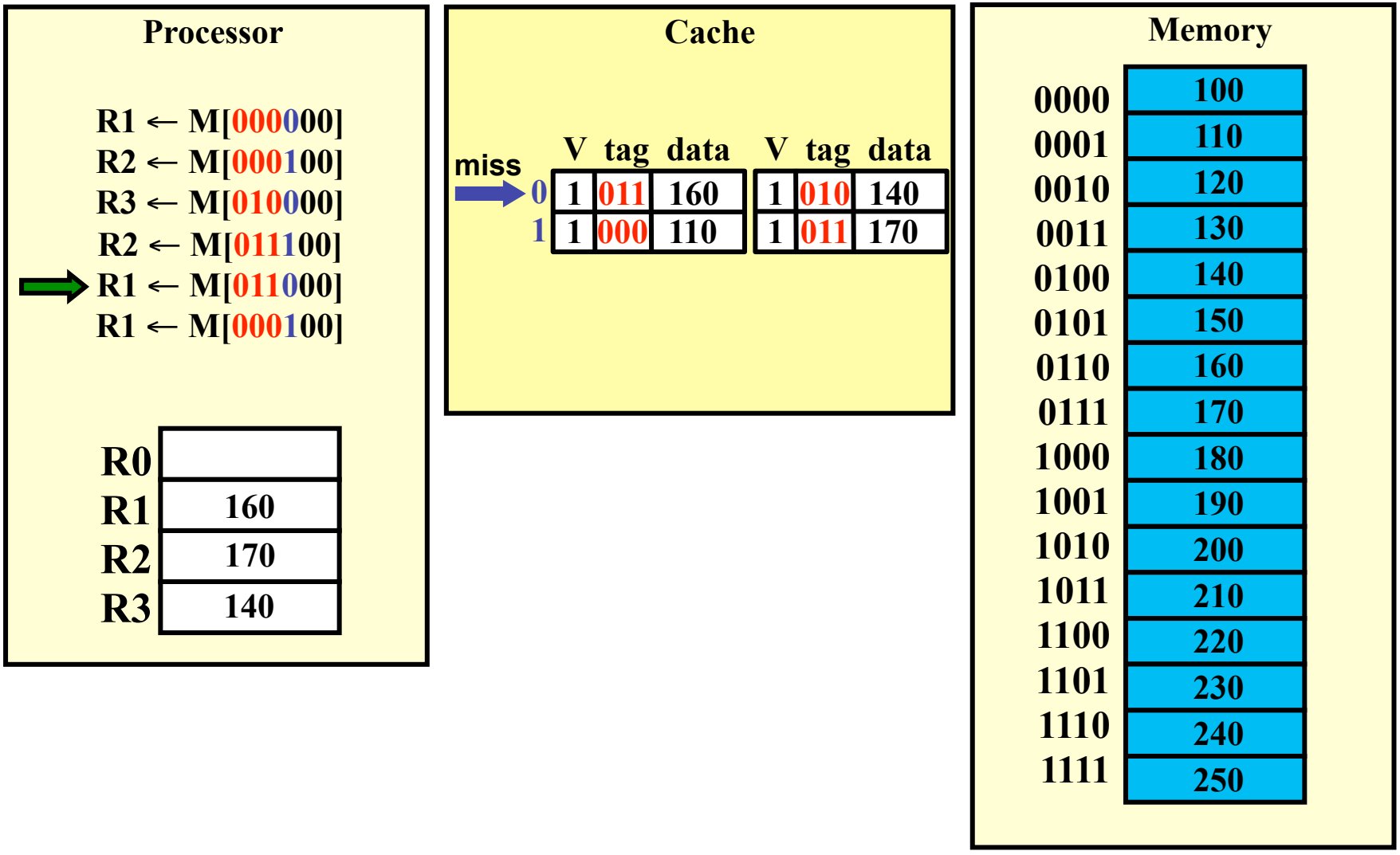
2-way Set Associative Example



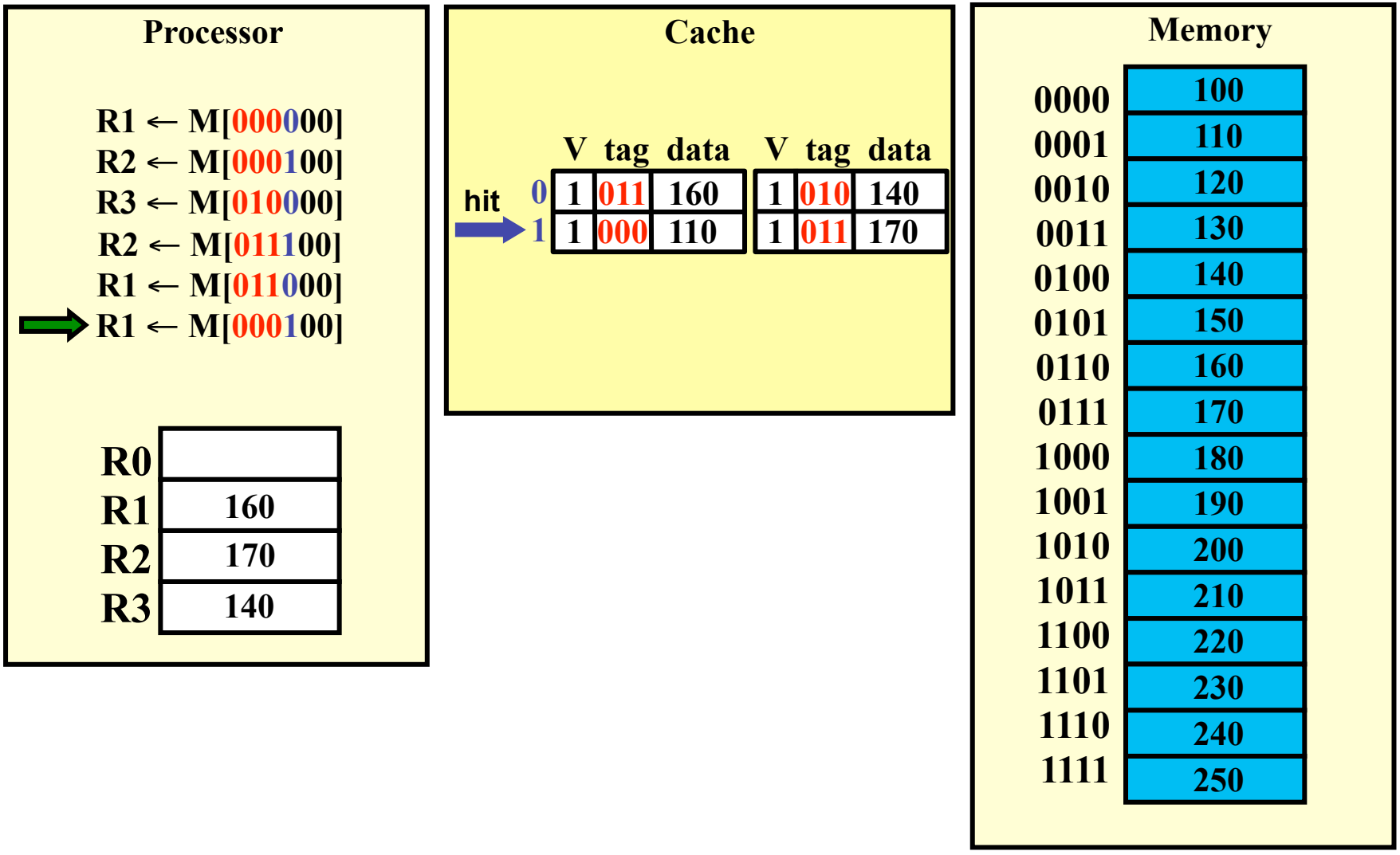
2-way Set Associative Example



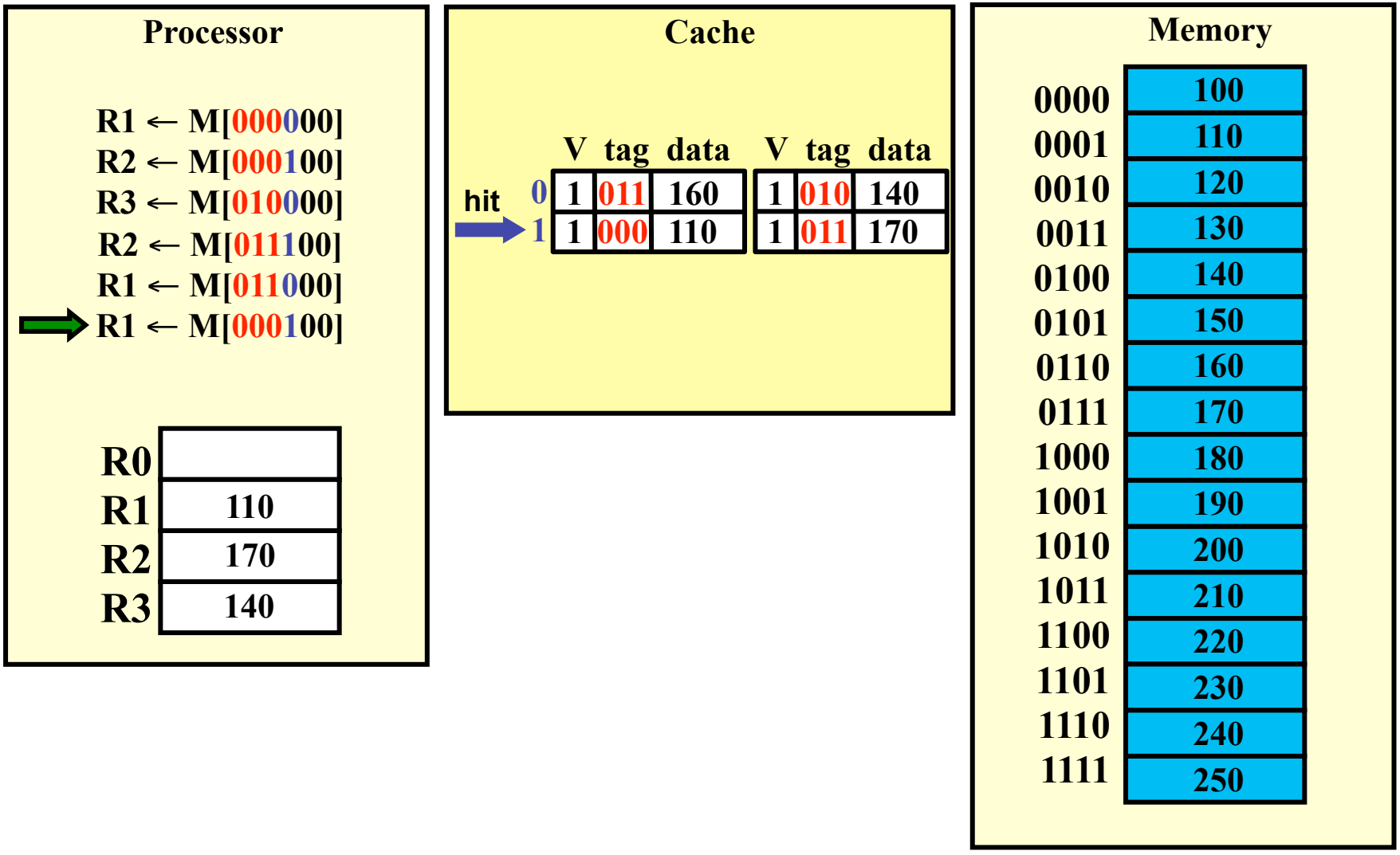
2-way Set Associative Example



2-way Set Associative Example



2-way Set Associative Example



Miss Classification

- Compulsory (Cold) misses
  - Caused by the first access to a block
- Capacity misses
  - Occur because the cache might not big big enough to hold all the blocks needed during execution
- Conflict misses
  - Occur in set-associative or direct mapped cache when multiple blocks compete for the same set, but would not occur in a fully associative cache of the same size

Next Time

More Caches