

## 18. Direct-mapped and associative caches

---

EECS 370 – Introduction to Computer Organization – Winter 2009

**Prof. Todd Austin & Prof. Marios Papaefthymiou**

EECS Department  
University of Michigan, Ann Arbor, USA

© Austin & Papaefthymiou, 2009

The material in this presentation cannot be  
copied in any form without our written permission

## Announcements

---

- ❑ Project 3
  - Due this Friday, March 27th
- ❑ Project 4
  - Caches
  - Will be available early next week...
- ❑ Exam 2
  - Thursday April 2, in class

## Associativity

---

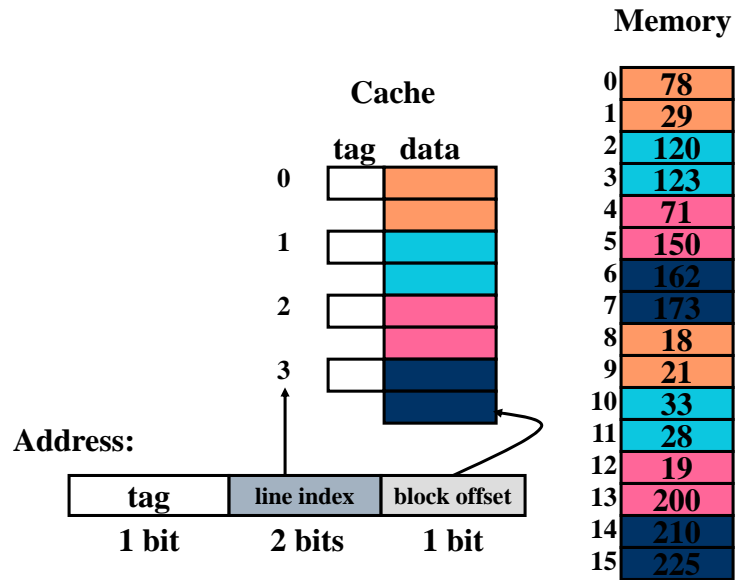
- ❑ Today, we will finish off caches.
- ❑ So far, we have designed a **fully associative** cache.
  - Any memory location can be copied to any cache line.
  - We check every cache tag to determine whether the data is in the cache.
- ❑ This approach can be too slow sometimes.
  - Parallel tag searches are slower. Why?

## Direct mapped cache

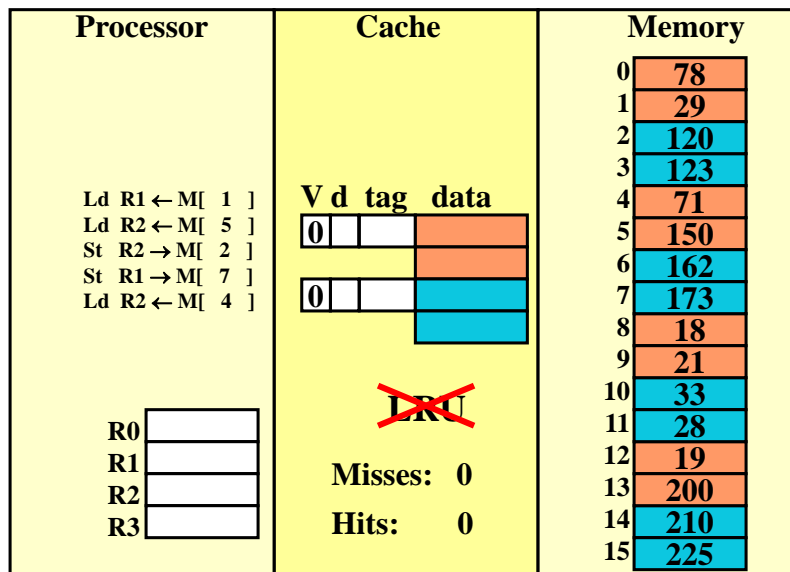
---

- ❑ We can redesign the cache to eliminate the requirement for parallel tag lookups.
  - **Direct mapped** caches partition memory into as many regions as there are cache lines.
  - Each memory region has a single cache line in which data can be placed.
  - You then only need to check a single tag – the one associated with the region the reference is located in.

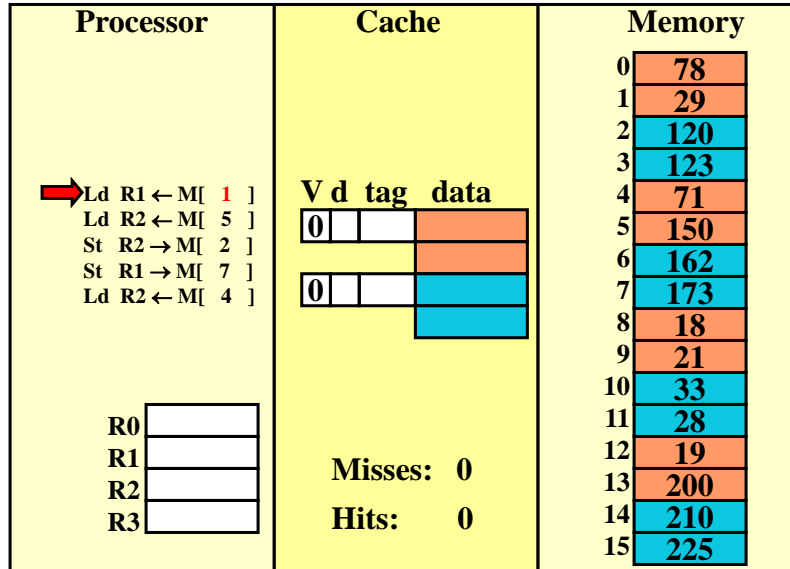
## Mapping memory to cache



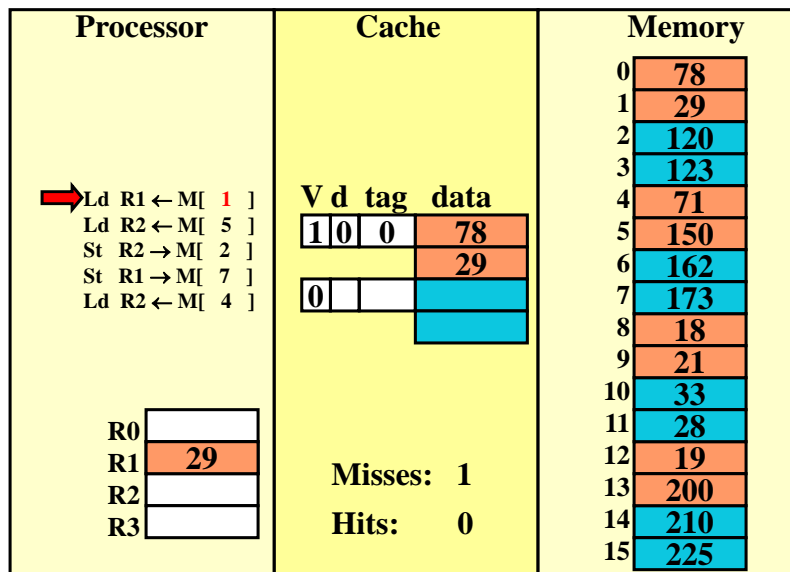
## Direct-mapped cache



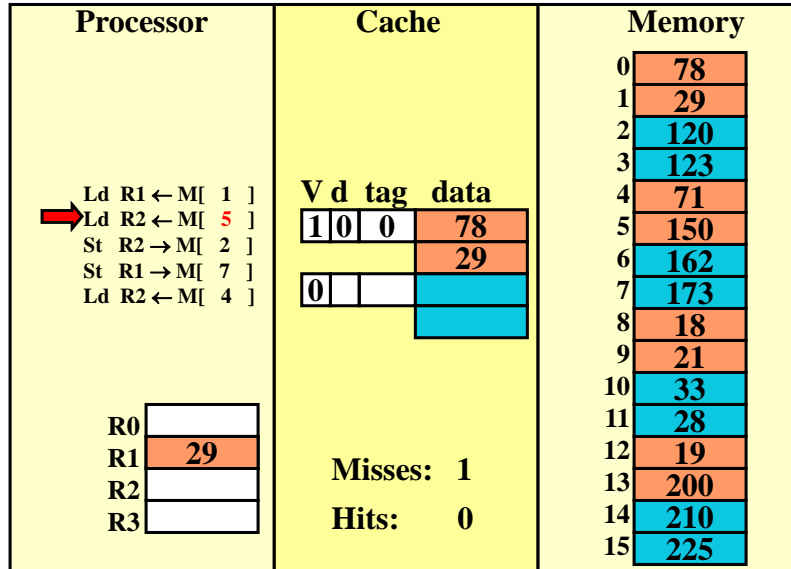
### Direct-mapped (REF 1)



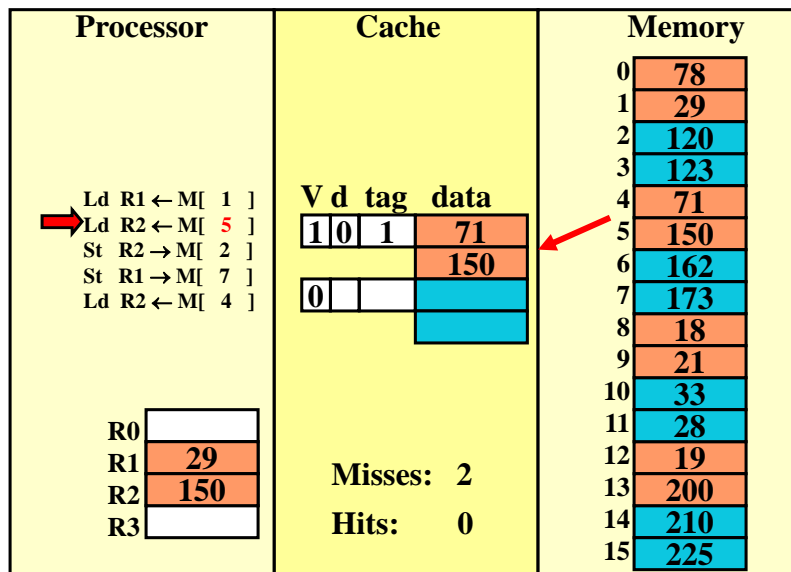
### Direct-mapped (REF 1)



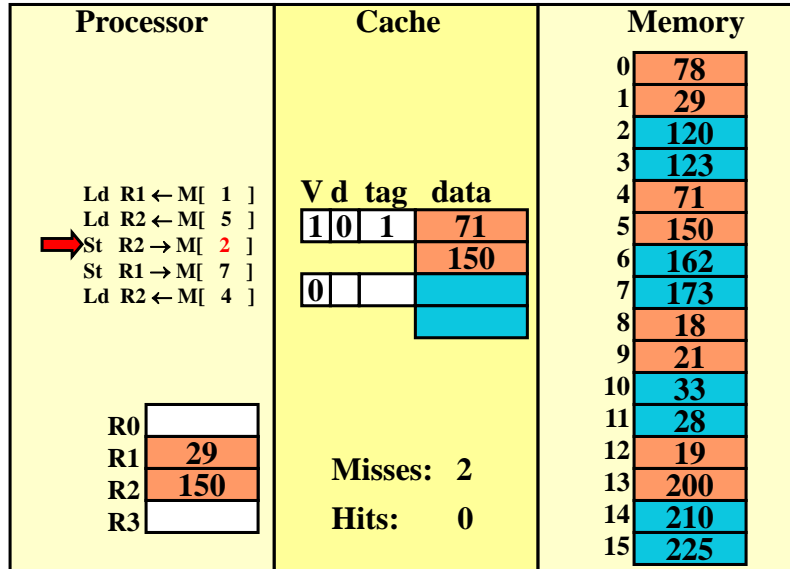
### Direct-mapped (REF 2)



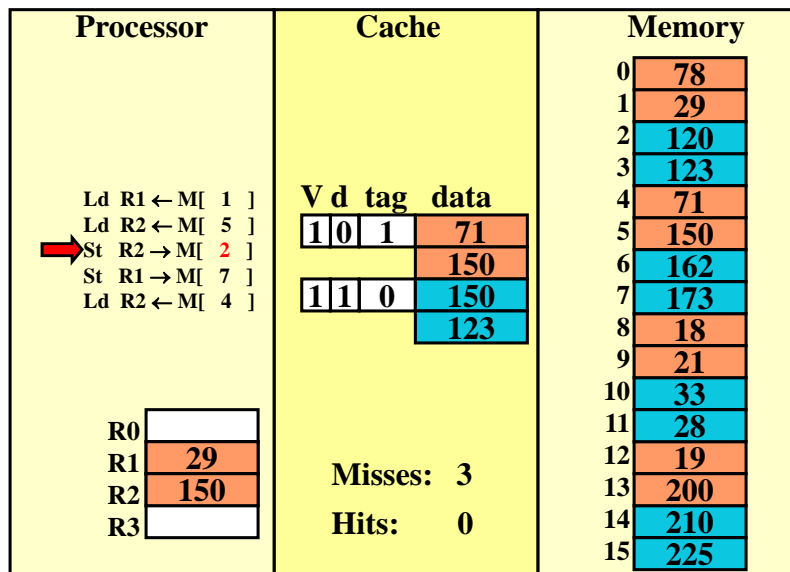
### Direct-mapped (REF 2)



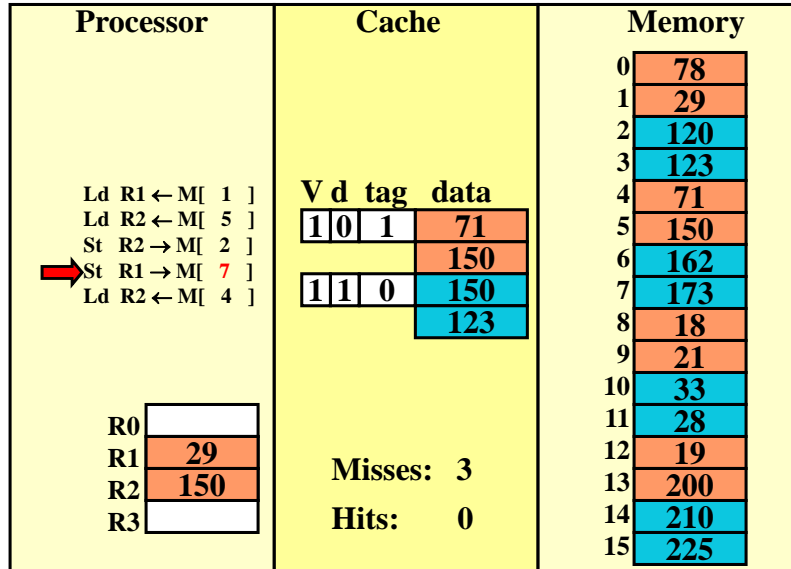
### Direct-mapped (REF 3)



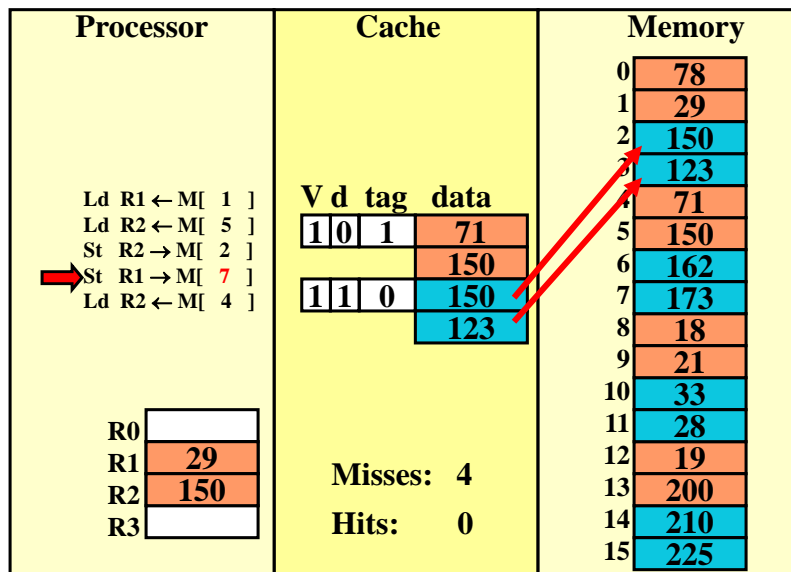
### Direct-mapped (REF 3)



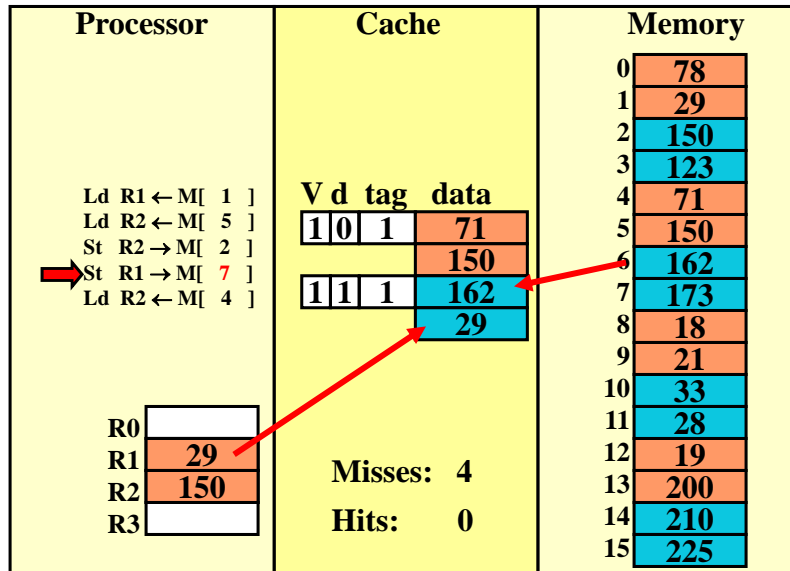
### Direct-mapped (REF 4)



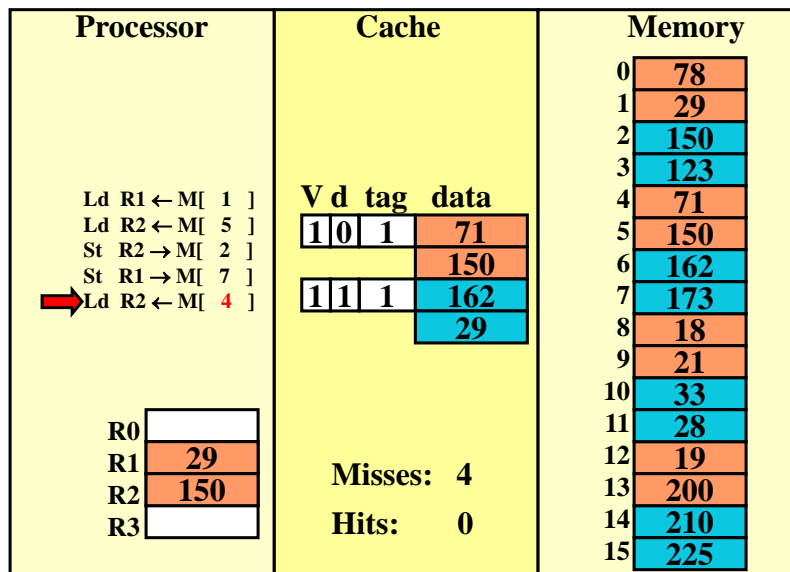
### Direct-mapped (REF 4)



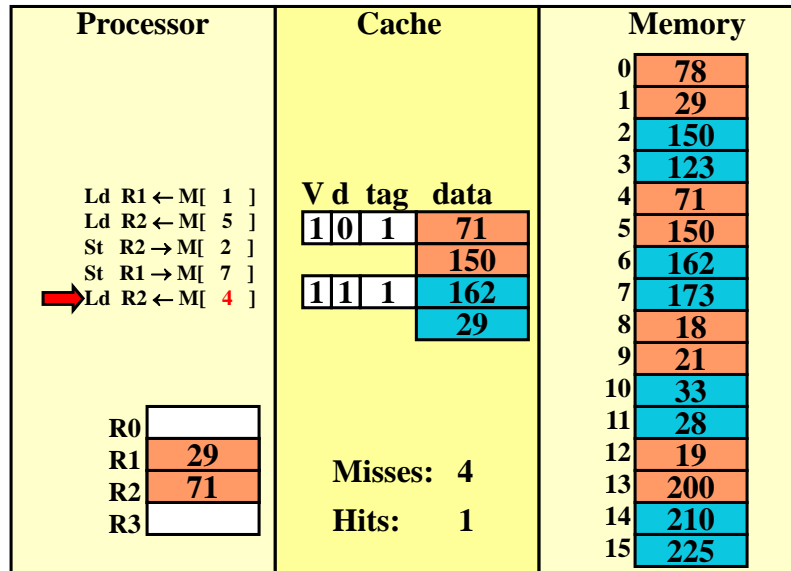
### Direct-mapped (REF 4)



### Direct-mapped (REF 5)



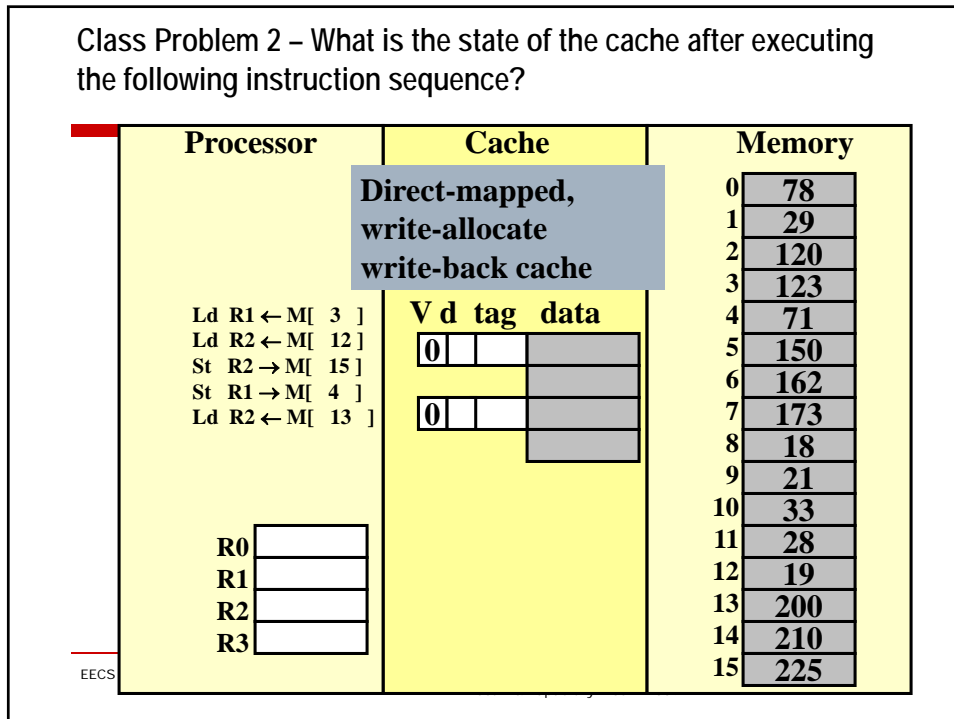
## Direct-mapped (REF 5)



## Class Problem 1

- How many tag bits are required for:
  - 32-bit address, byte addressed, direct-mapped 32k cache, 128 byte block size
  - 16-bit address, word addressed, direct mapped 1k word cache, 16 word block size

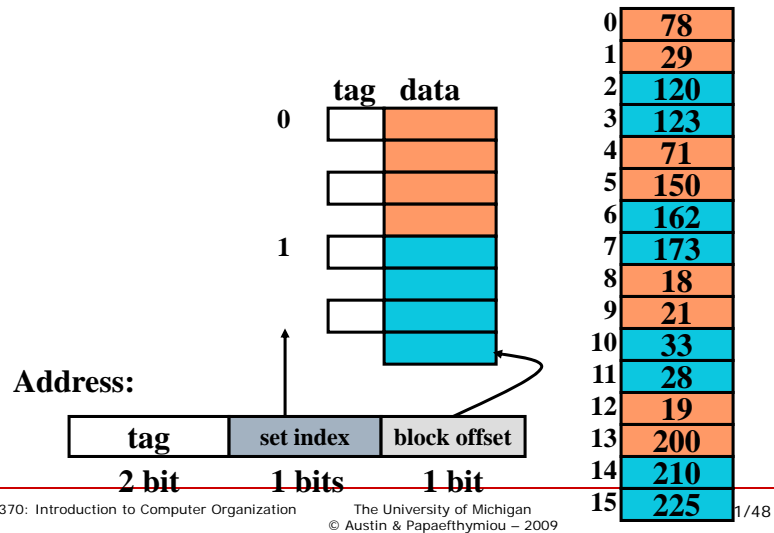
Class Problem 2 – What is the state of the cache after executing the following instruction sequence?



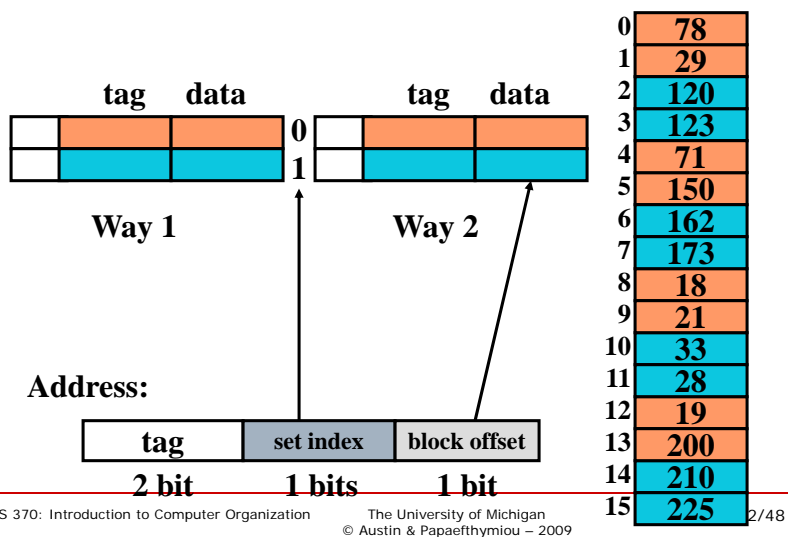
### Split the difference

- ❑ **Set associative** caches:
  - Partition memory into regions
    - like direct mapped but fewer partitions
  - Associate a region to a **set** of cache lines
    - Check tags for all lines in a set to determine a HIT
- ❑ Treat each line in a set like a small fully associative cache.
  - LRU (or LRU-like) policy generally used.

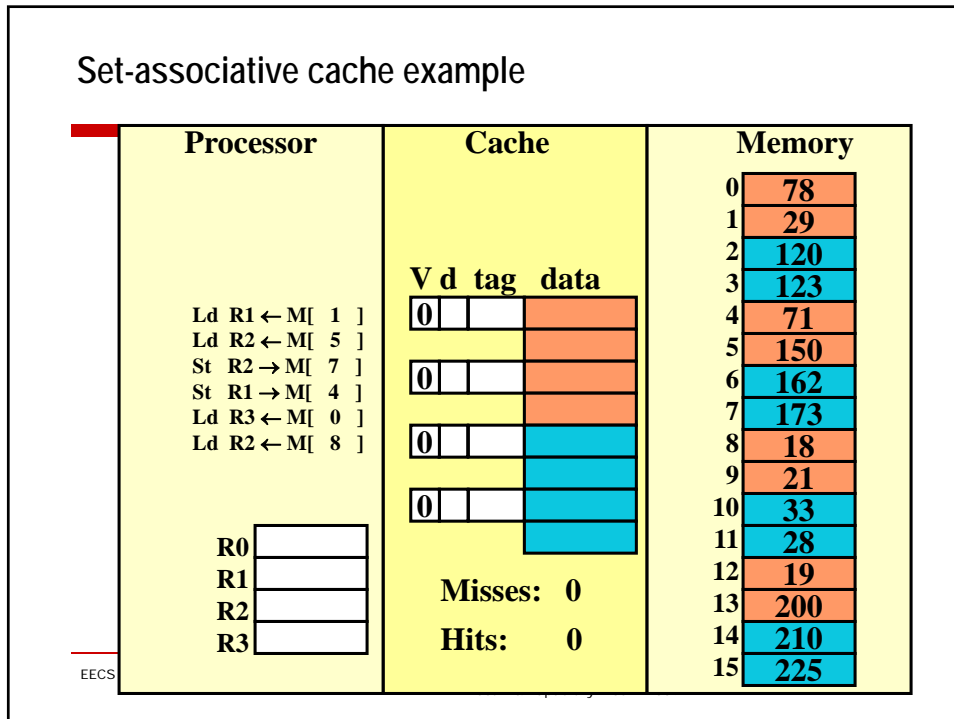
## Set Associative Cache



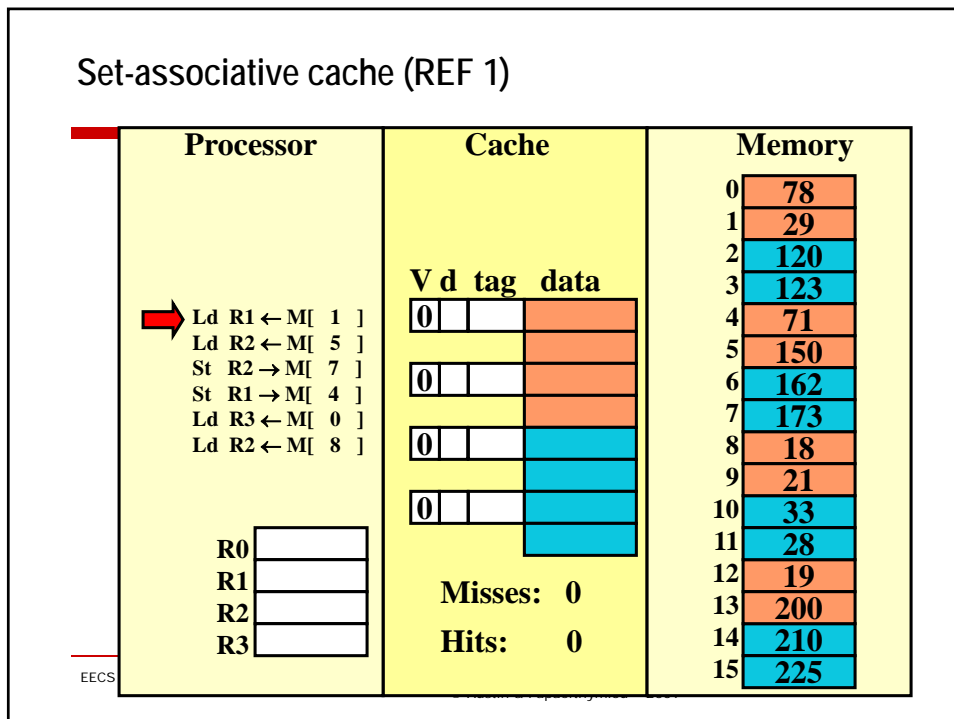
## Set Associative Cache using the book's style



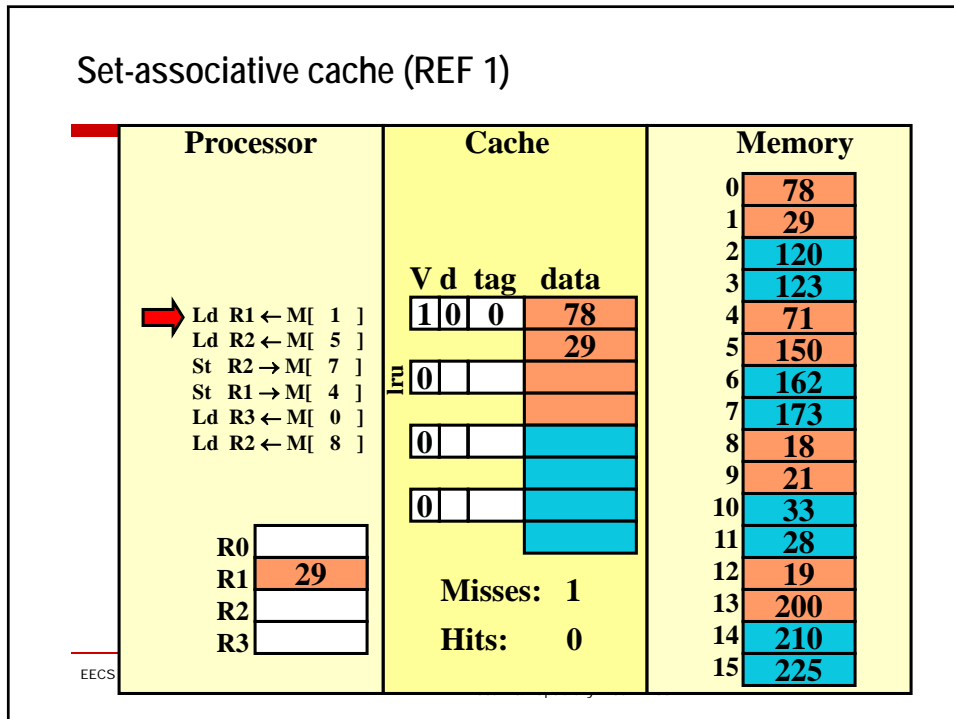
### Set-associative cache example



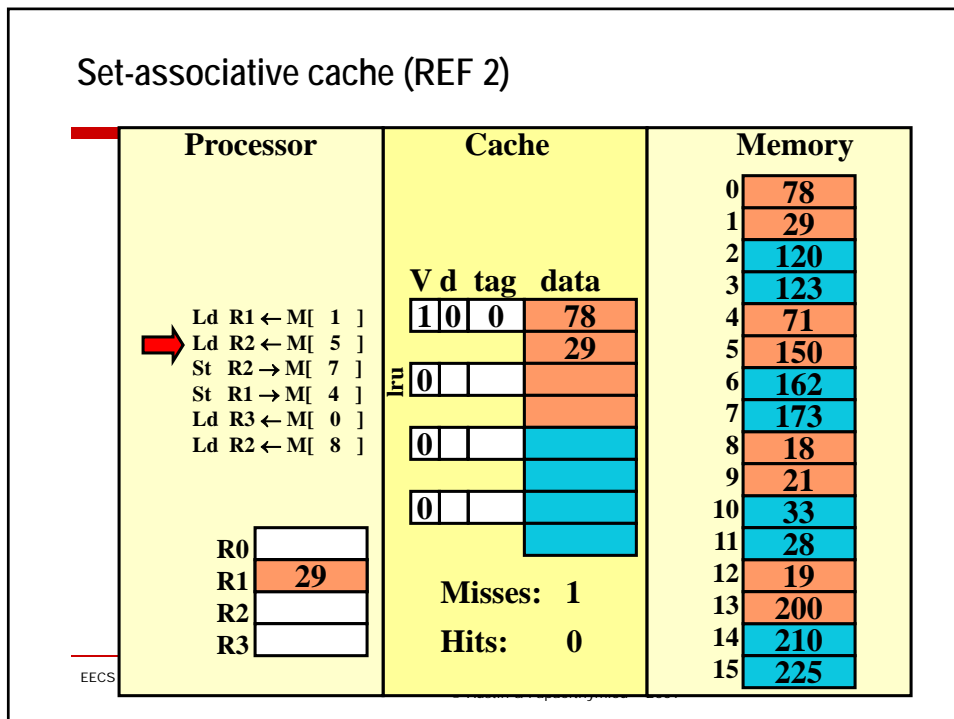
### Set-associative cache (REF 1)



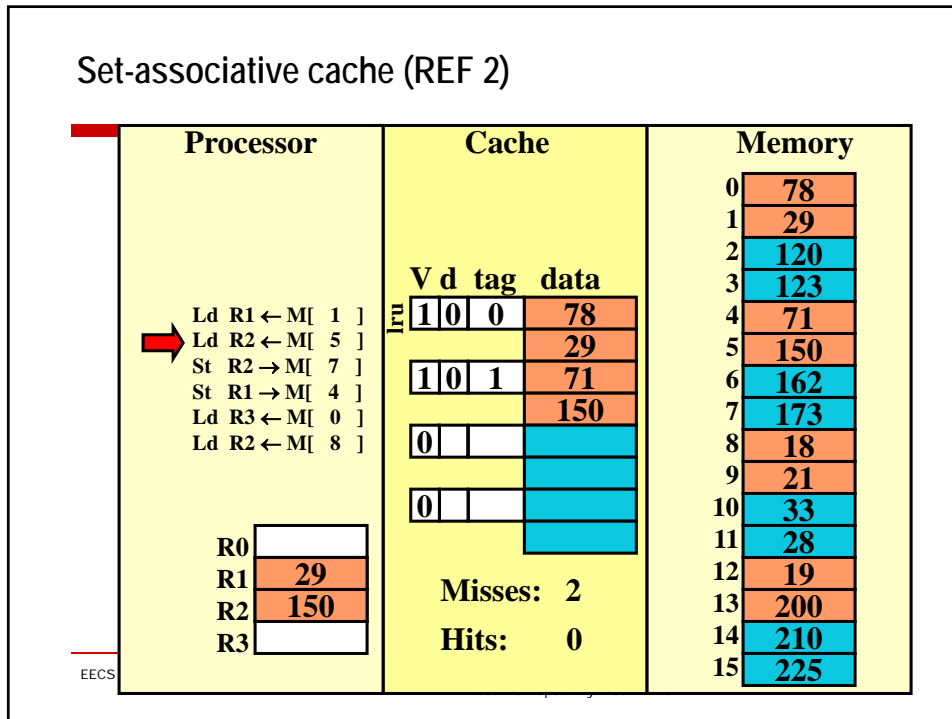
### Set-associative cache (REF 1)



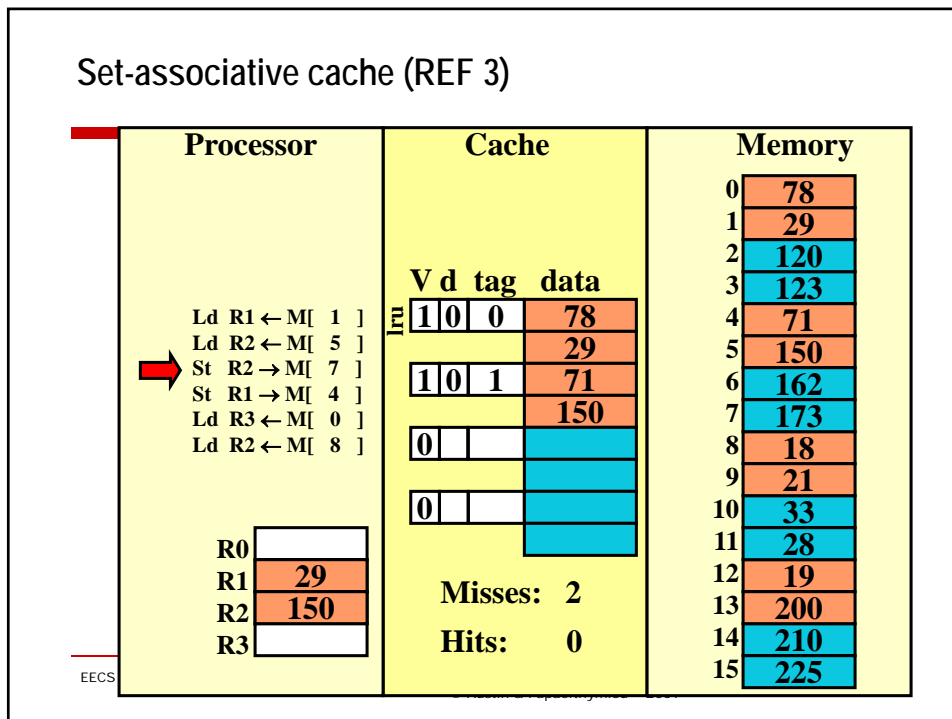
### Set-associative cache (REF 2)



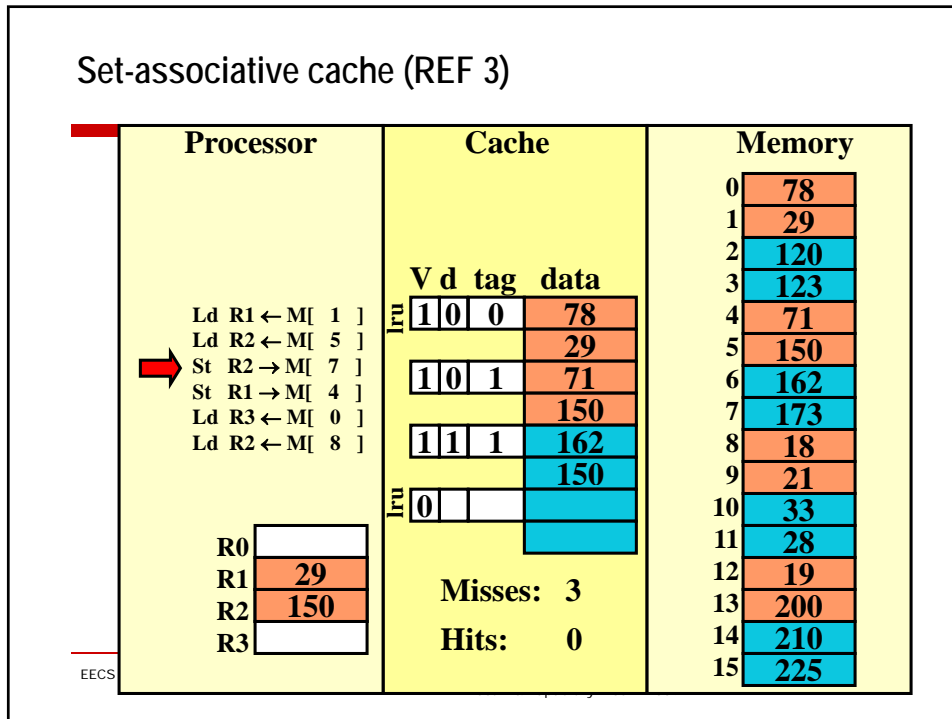
### Set-associative cache (REF 2)



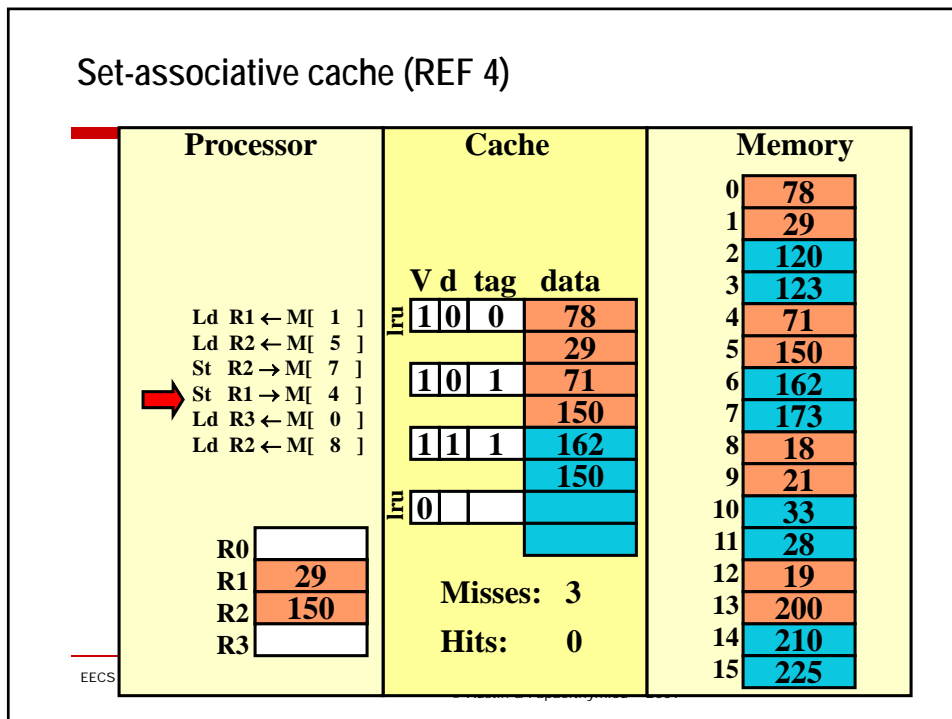
### Set-associative cache (REF 3)



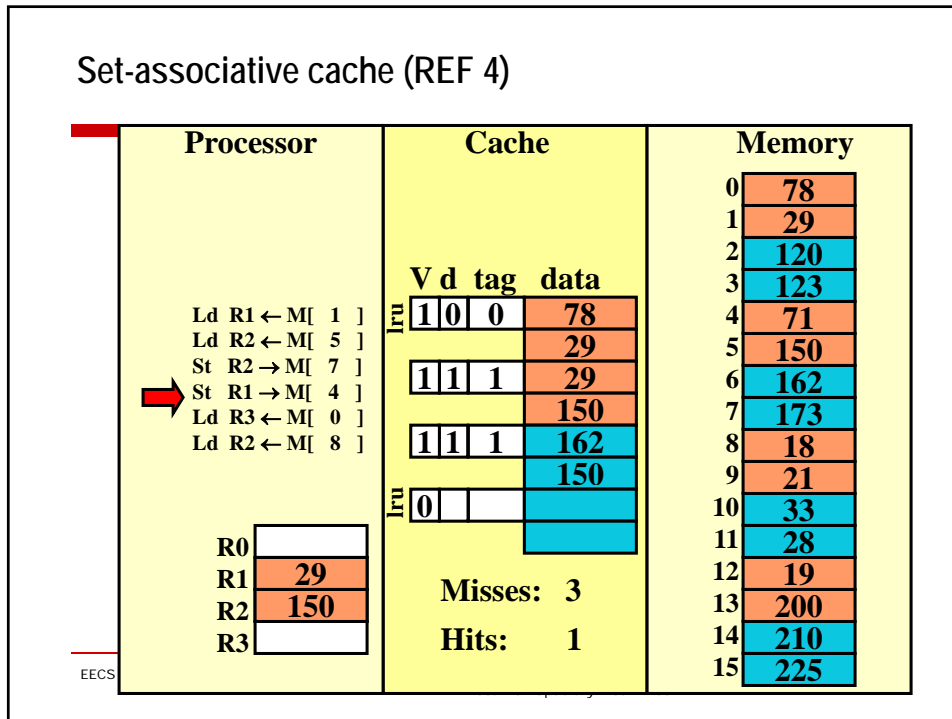
### Set-associative cache (REF 3)



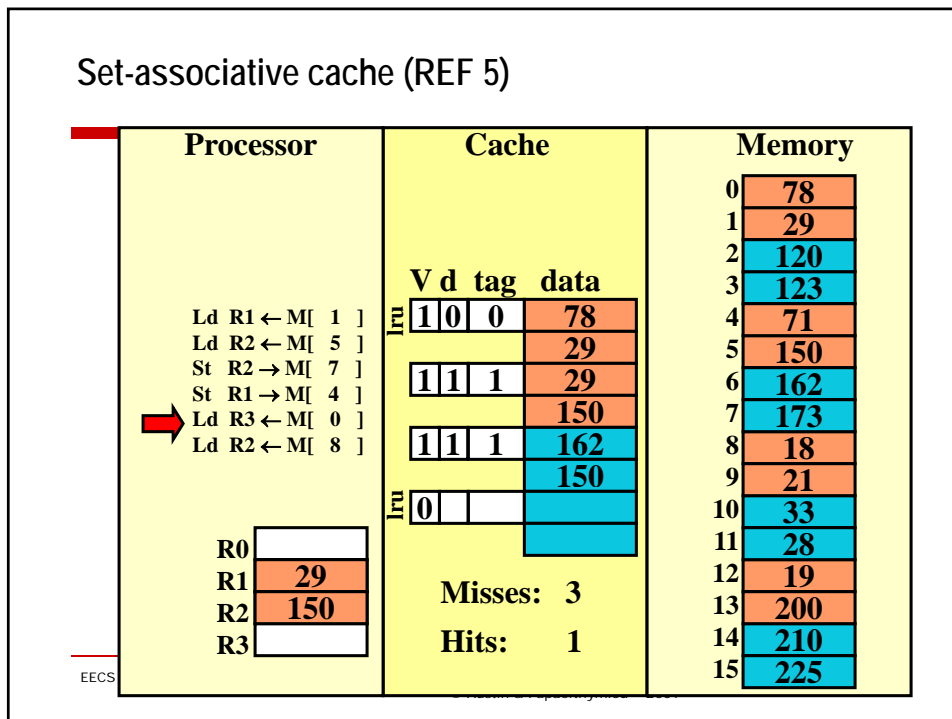
### Set-associative cache (REF 4)



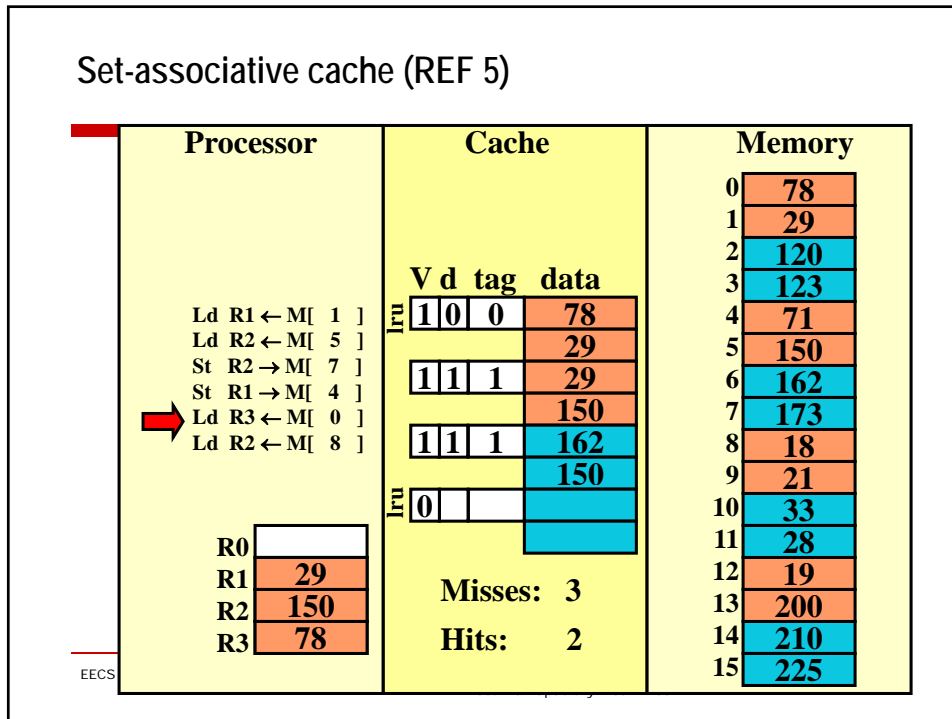
### Set-associative cache (REF 4)



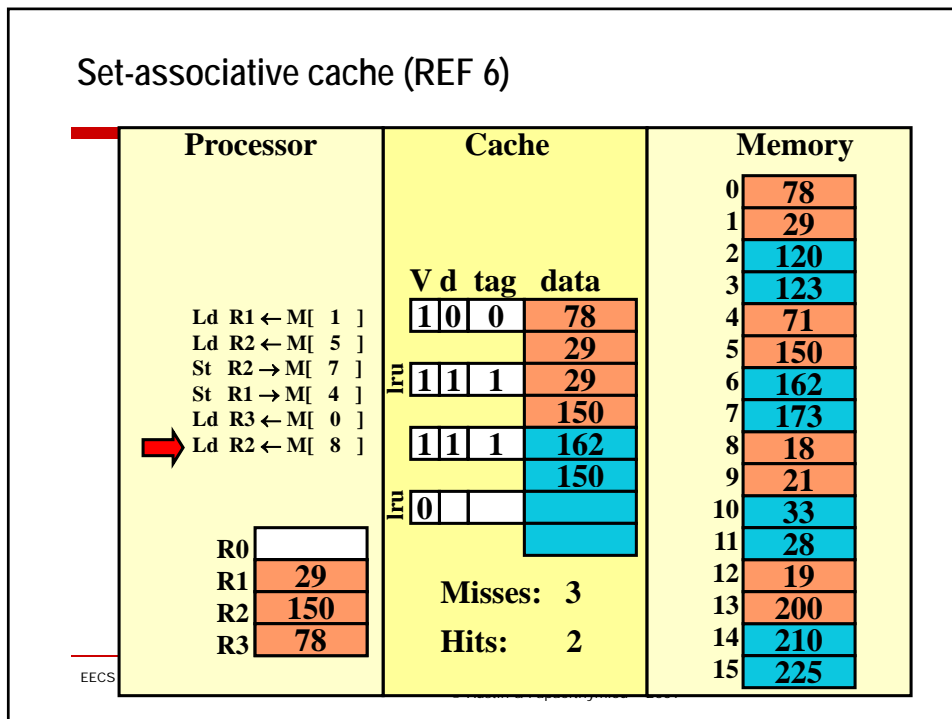
### Set-associative cache (REF 5)



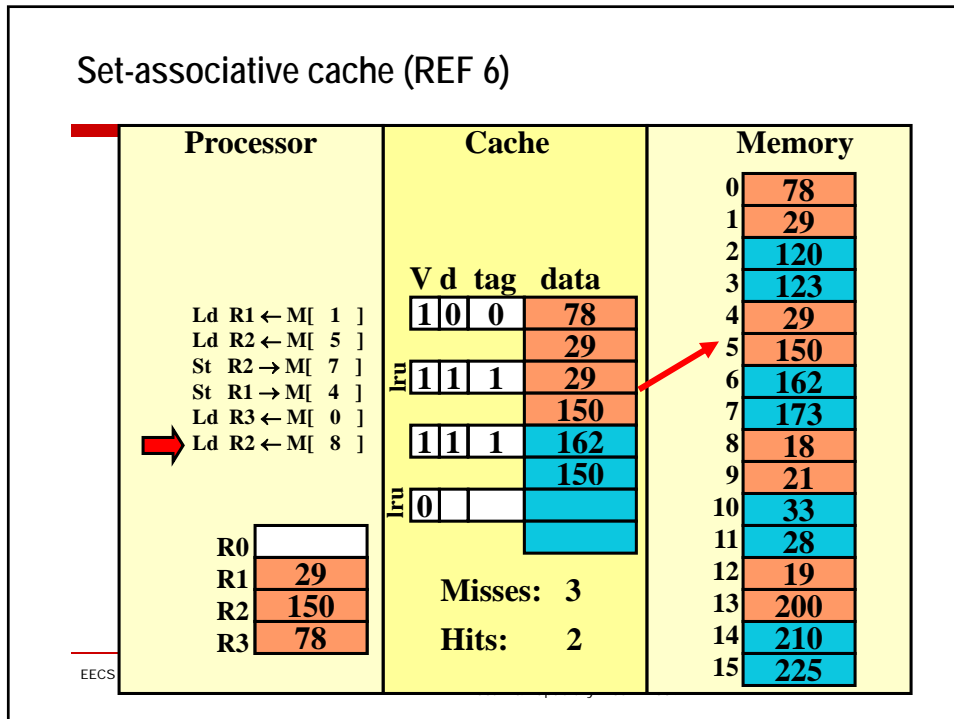
### Set-associative cache (REF 5)



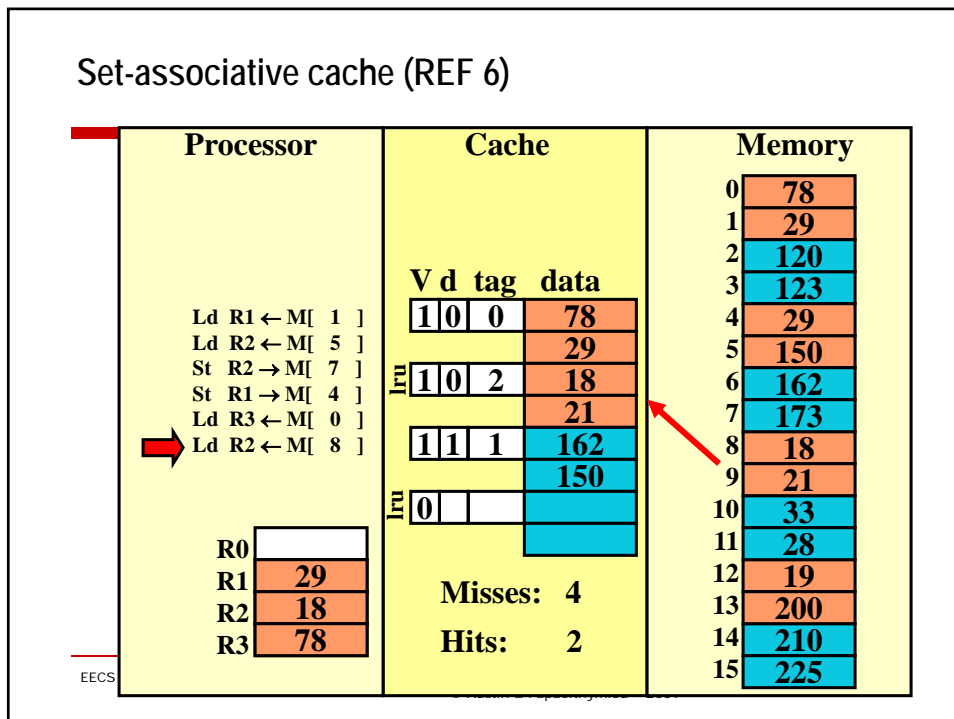
### Set-associative cache (REF 6)



### Set-associative cache (REF 6)



### Set-associative cache (REF 6)



## Cache Organization Comparison

Block size = 2 bytes, total cache size = 8 bytes for all caches

### 1. Fully associative (4-way associative)

V d tag data



### 2. Direct mapped

V d tag data



### 3. 2-way associative

V d tag data



## Reasons for cache misses A.K.A. the 3 C's of cache misses

- ❑ First reference to an address
  - **Compulsory** miss
    - Also sometimes called a “cold start” miss
    - First reference to any block will always miss
    - Reduce by **increasing block size** (to reduce number of blocks)
- ❑ Cache is too small to hold all the data
  - **Capacity** miss
    - Would have had a hit with a large enough cache
    - Reduce by **building a bigger cache**
- ❑ Replaced it from a busy set
  - **Conflict** miss
    - Would have had a hit with a fully associative cache
    - Reduce by **increasing associativity**

## Classifying cache misses

---

- ❑ Can you classify all cache misses?
  - Compulsory miss?
  - Capacity miss?
  - Conflict miss?
- ❑ Yes! (with a cache simulator)
  - Simulate with a cache of unlimited size (cache size = memory size)
    - Any misses must be compulsory misses
  - Simulate again with a fully associative cache of the intended size
    - Any new misses must be capacity misses
  - Simulate a third time, with the actual intended cache
    - Any new misses must be conflict misses

## Fixing cache misses

---

- ❑ **Capacity** misses
  - Would have had a hit with a large enough cache
  - Cache is not big enough
  - Reduce by building a bigger cache
- ❑ **Conflict** misses
  - Would have had a hit with a fully associative cache
  - Cache does not have enough associativity
  - Reduce by increasing associativity
- ❑ **Compulsory** misses
  - First reference to any block will always miss
  - No way to completely avoid these
  - Reduce by increasing block size
  - This reduces the total number of blocks

## What about Instructions?

---

- ❑ Instructions should be cached as well.
- ❑ We have two choices:
  1. Treat instruction fetches as normal data and allocate cache lines when fetched.
  2. Create a second cache (called the **instruction cache** or **ICache**) which caches instructions only.
    - How do you know which cache to use?
    - What are advantages of a separate ICache?

## Integrating Caches into a Pipeline

---

- ❑ How are caches integrated into a pipelined implementation?
  - Replace instruction memory with Icache.
  - Replace data memory with Dcache.
- ❑ Issues
  - Memory accesses now have variable latency.
  - Both caches may miss at the same time.



## Class Problem 2

---

Show the breakdown of the address for the following cache configuration:

**32 bit address**

**16K cache**

**4-way set associative cache**

**64-byte blocks**

## Class Problem 3

---

Show the breakdown of the address for the following cache configuration:

**32 bit address**

**16K cache**

**Direct-mapped cache**

**32-byte blocks**

## Class Problem 4

---

Given the following cache configuration:

Cache size = 16 bytes	Write allocate
Block size = 2 bytes	LRU replacement policy
Direct mapped	Address size = 16 bits, byte addressable memory

For the following sequence of addresses, label each reference as a hit (H) or miss (M). For the misses, indicate the type (conflict, capacity, compulsory).

4 7 21 22 23 20 5 7 36 6 4

## Class Problem 5

---

Given the following: instruction breakdown:  
LW 25%, Add 20%, BEQ 20%, SW, 15%, Nand 20%;  
branches are taken 60% of the time, LW followed by  
immediate use occurs 20% of the time. The instruction  
cache hit rate is 90% and the data cache hit rate is 80%.  
Assume each cache miss requires 10 stall cycles.  
Data hazards – detect and forward  
Control hazards – predict not taken and squash

Calculate the CPI