Multicolumn Cache: aiming for both high hit rate and low latency

Xiaodong Zhang

CSE 3421 additional notes for Memory Systems

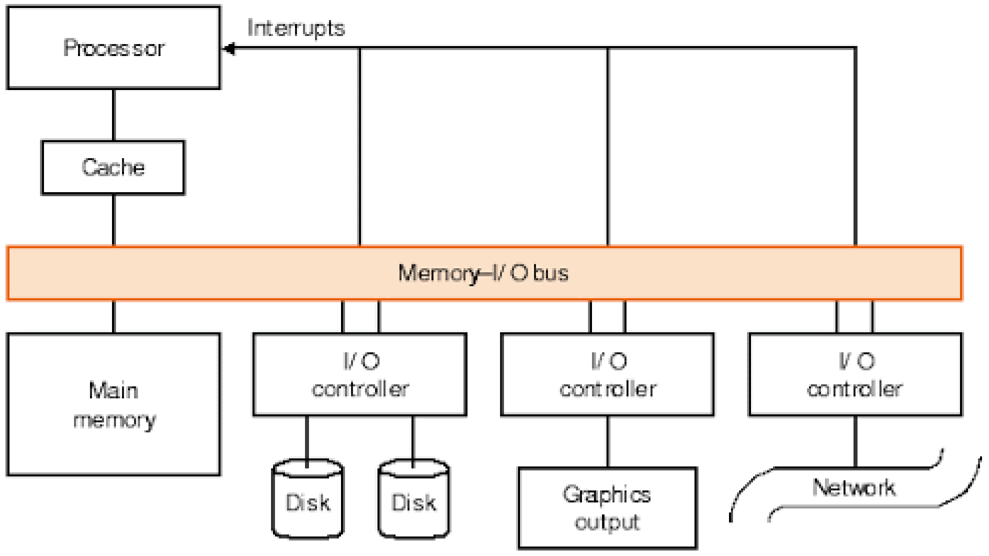
Basic concepts of hardware cache

- Cache design is memory-address centric
  - The default length is 32 bits, each memory address is unique and points to a byte (byte addressable)
  - Each memory address connects to the first byte of the content, and its default length is a word (4 Bytes or 32 bits)

- Cache is a small storage inclusively built in the CPU chip
  - It contains copies of the data from memory (and disks)
  - A cache-block storage can be multiple-words long
  - Besides data storage, each block needs tags V bit, and others

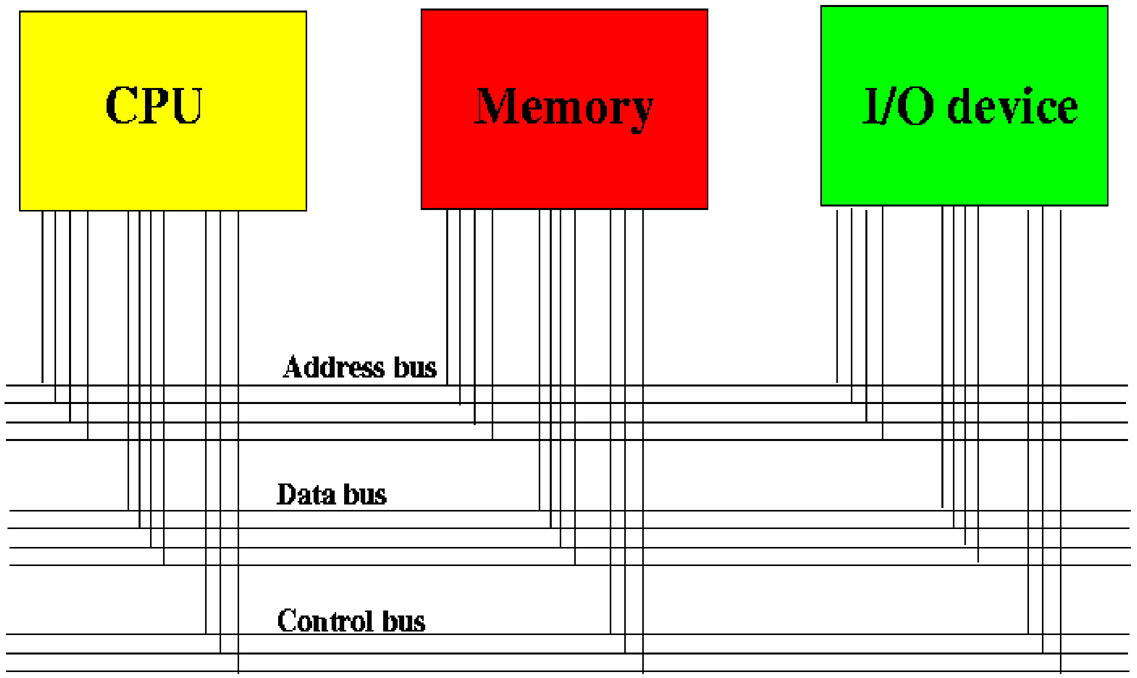
- For a given memory address, CPU knows where it is
  - By direct, set-associative or fully associative mapping
  - How do handle write hit, write miss, read hit, and read miss?
  - How to address the conflicts between high hit rates and low latency?

Connections between CPU-Caches and DRAM memory



Three steps for a cache miss:

- Sending a memory address (bus)
- Loading data from memory (no bus)
- Transferring data to CPU (bus)

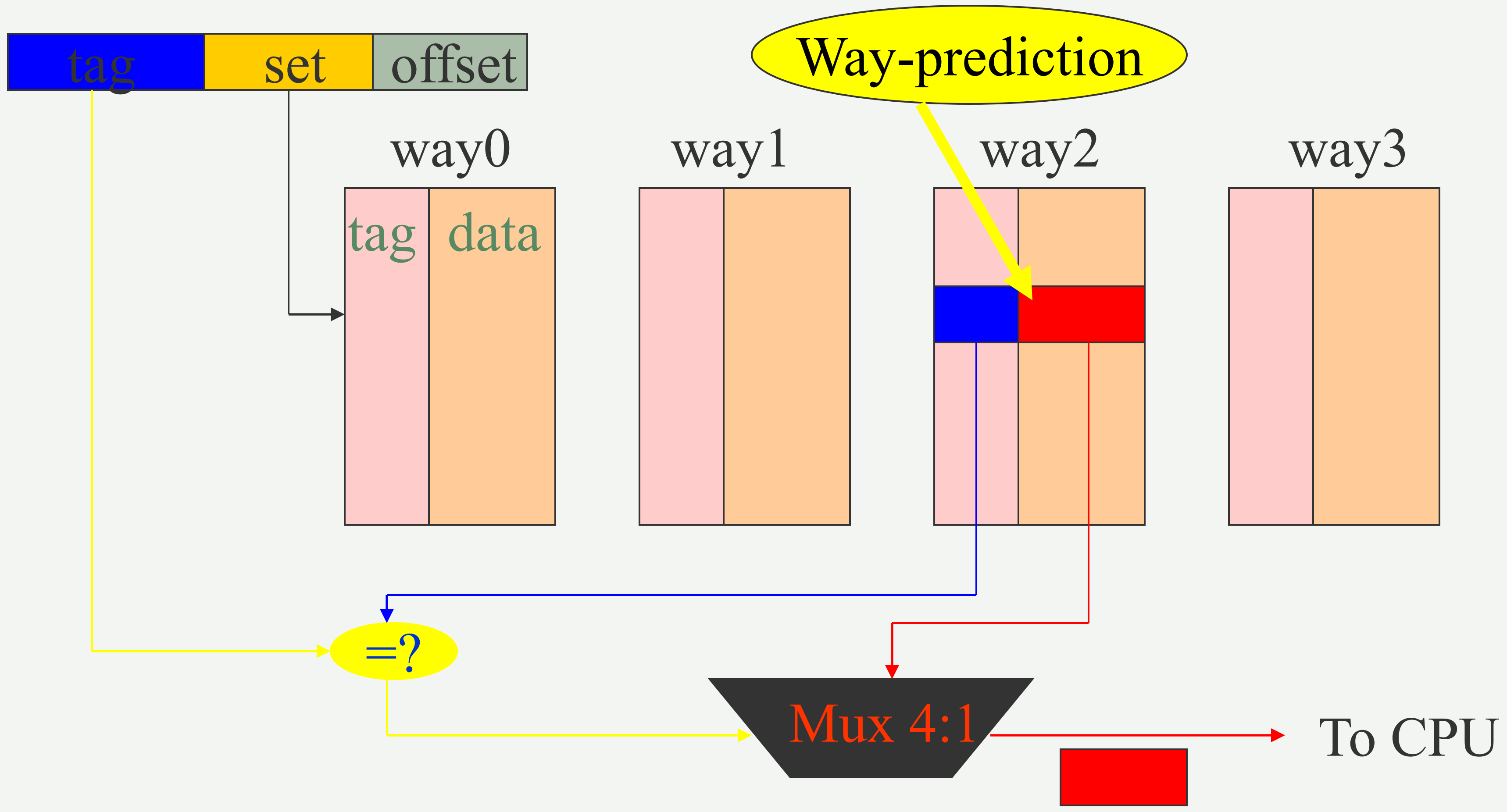


Trade-offs between High Hit Ratios and Low Latency

- Set- associative cache achieves high hit-ratios: 30% higher than that of direct-mapped cache.
- But it suffers high access times due to
  - Multiplexing logic delay during the selection.
  - Tag checking, selection, and data dispatching are sequential.
- Direct-mapped cache loads data and checks tag in parallel: minimizing the access time.
- High power consumption is another concern for set associative cache
- Can we get both high hit ratios and low latency in low power?

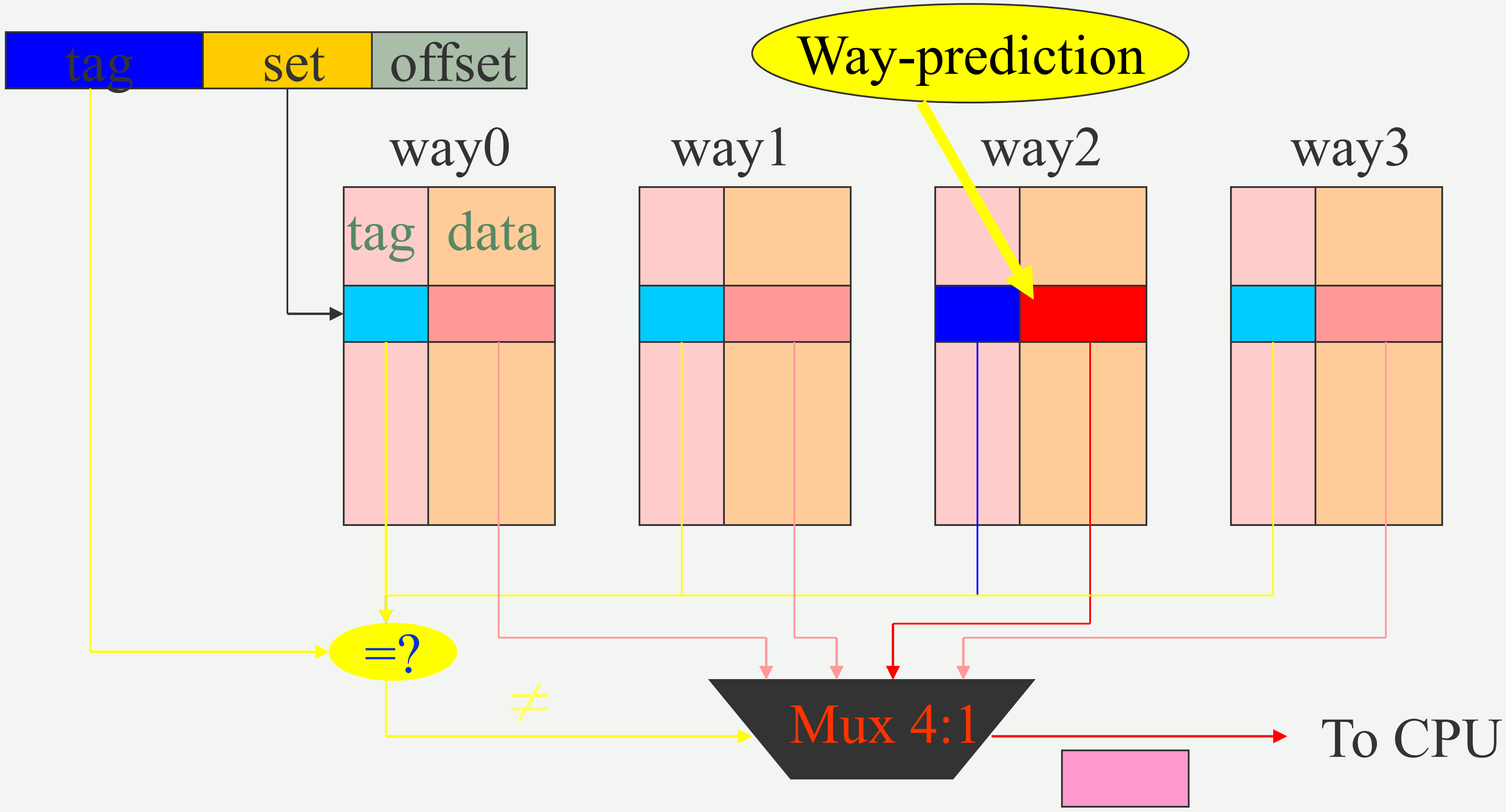
Best Case of Way Prediction: First Hit

Cost: way prediction only



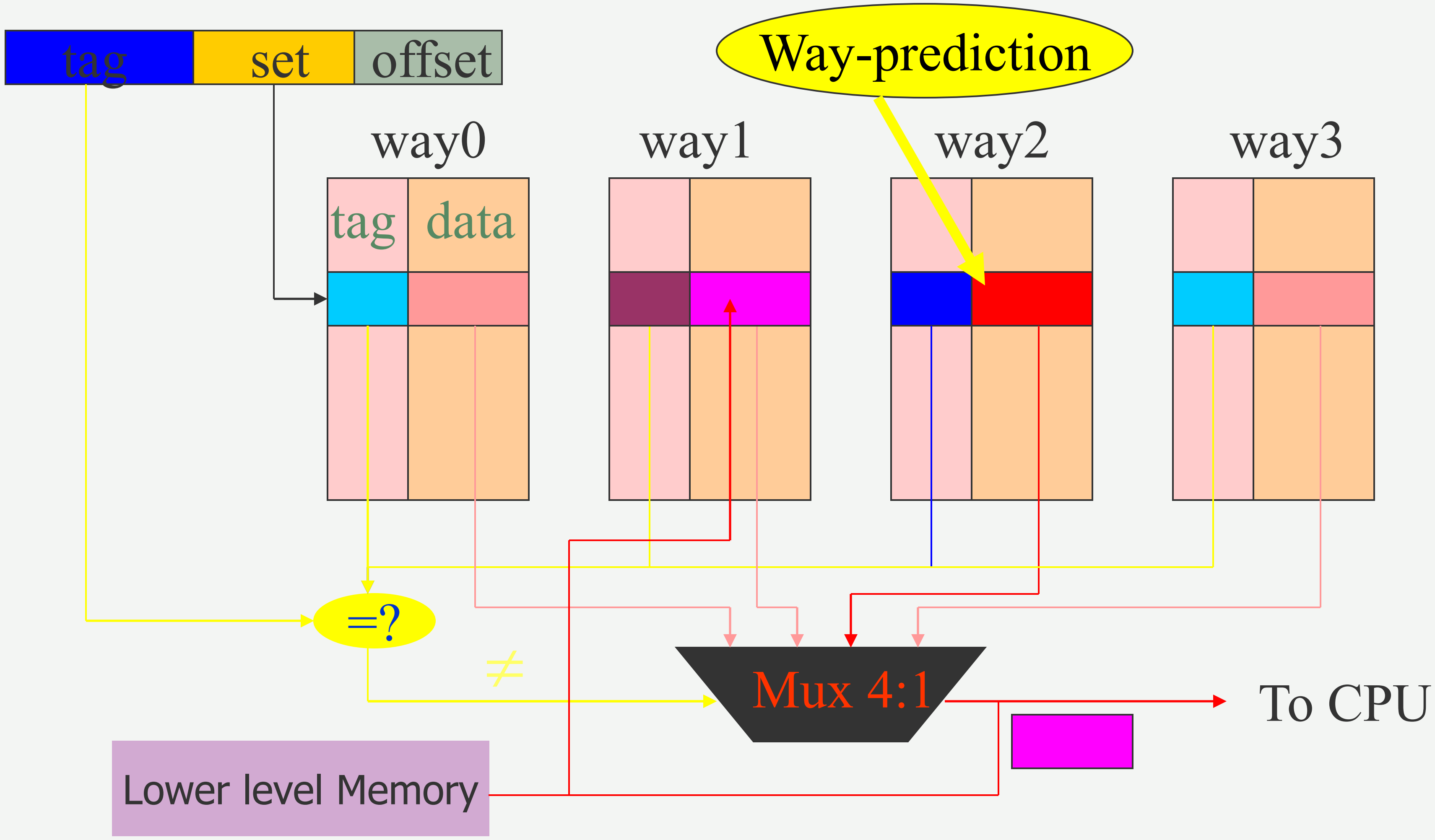
Way Prediction: Non-first Hit (in Set)

Cost: way prediction + selection in set



Worst Case of Way-prediction: Miss

Cost: way prediction + selection in set + miss



Make a low-latency and high hit-rate cache to be a reality

- A write to a set-associative cache
  1. Map to the cache set
  2. Find an empty way (block) to write
  3. Or write to the least recent used (LRU) way (block)
- A read in a set-associative cache
  1. Map to the cache set
  2. Compare with the tags, then select the way if matches
  3. Otherwise, serve the cache miss by going to memory
- Why a way-prediction is hard?
  - We do not know which way the cache block is located
  - Can we enforce which way to go in both write and read?

Basic Ideas

- Conventional Set Associative Cache
  - **Mapping** to a set of multiple ways
  - find a way (LRU) to **write**
  - search all ways for a **read**
  - **Latency** comes from **“finding”** or **“searching”** among **ways**

- Can we decisively or directly find a way?
  - If so, the search would be direct to the way

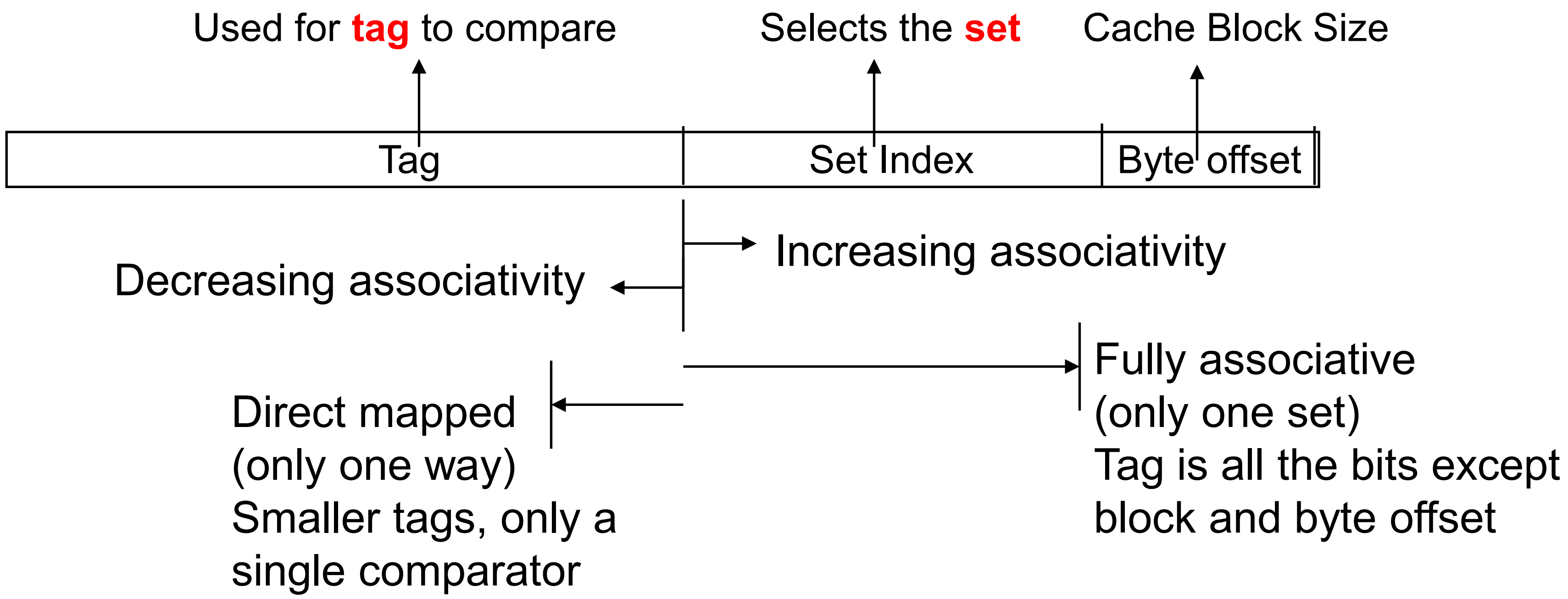
- How do we make this happen?
  - Multi-column cache

Multi-column Caches: Fast and High Hit Ratio

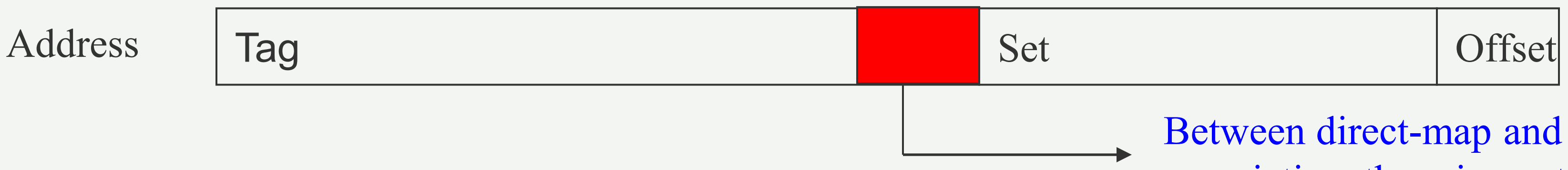
- Zhang, et. al., IEEE Micro, 1997.
- Objectives:
  - Each first hit is equivalent to a direct-mapped access.
  - Maximize the number of first hits.
  - Minimize the latency of non-first-hits.
  - Additional hardware should be simple and low cost.

Range of Set Associative Caches

- For a fixed size cache, each time the associativity increases it doubles the number of blocks per set (or the number of ways)
  - halves the number of sets – decreases the set index by 1 bit
  - increases the size of the tag by 1 bit
- Note: word offset is for multiple words in a block. word offset + byte offset = long cache block



Multi-Column Caches: Major Location

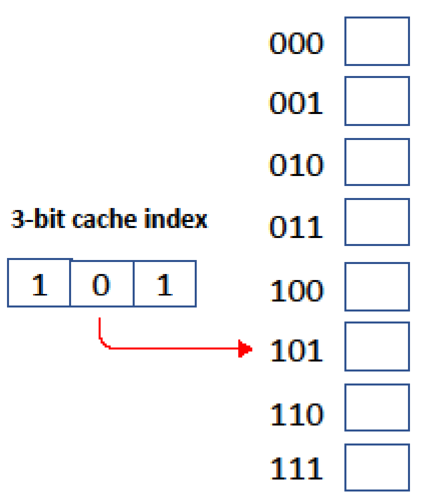


Between direct-map and set-associative, there is a set of unused bits for tags.

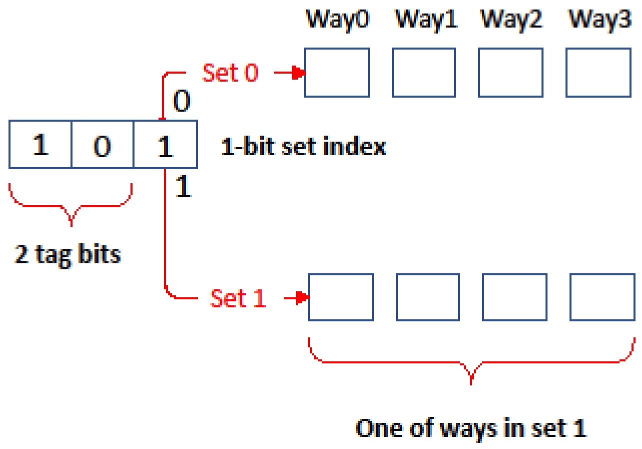
Let’s use the bits direct-mapping to the block (way) in the set

- The bits given to tag and ``set bits” generate a direct-mapped location: Major Location.
- A major location mapping = direct-mapping.

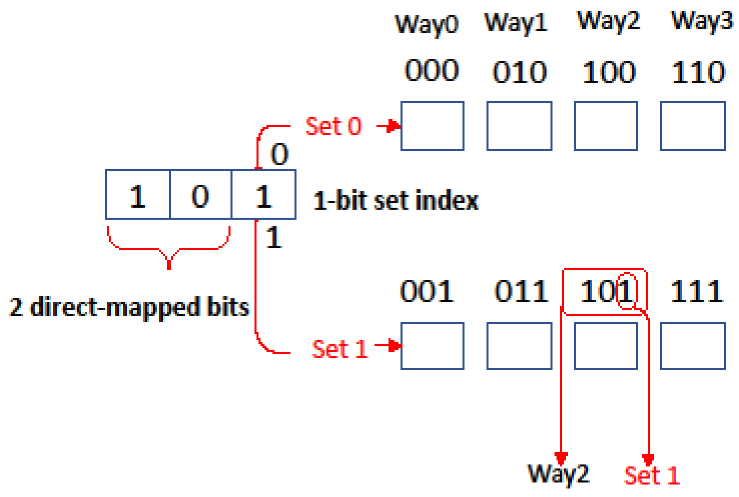
Comparisons of Three Caches



(a)



(b)



(c)

Implicit direct mapping in set associative cache

- Standard set associative cache only maps to a set

- Multicolumn cache uses the full direct map bits to write and read in a set associative cache
  - Set index remains the same
  - The bits given to the tag from direct map are used for which way in the set
  - These two sets of bits are available and free

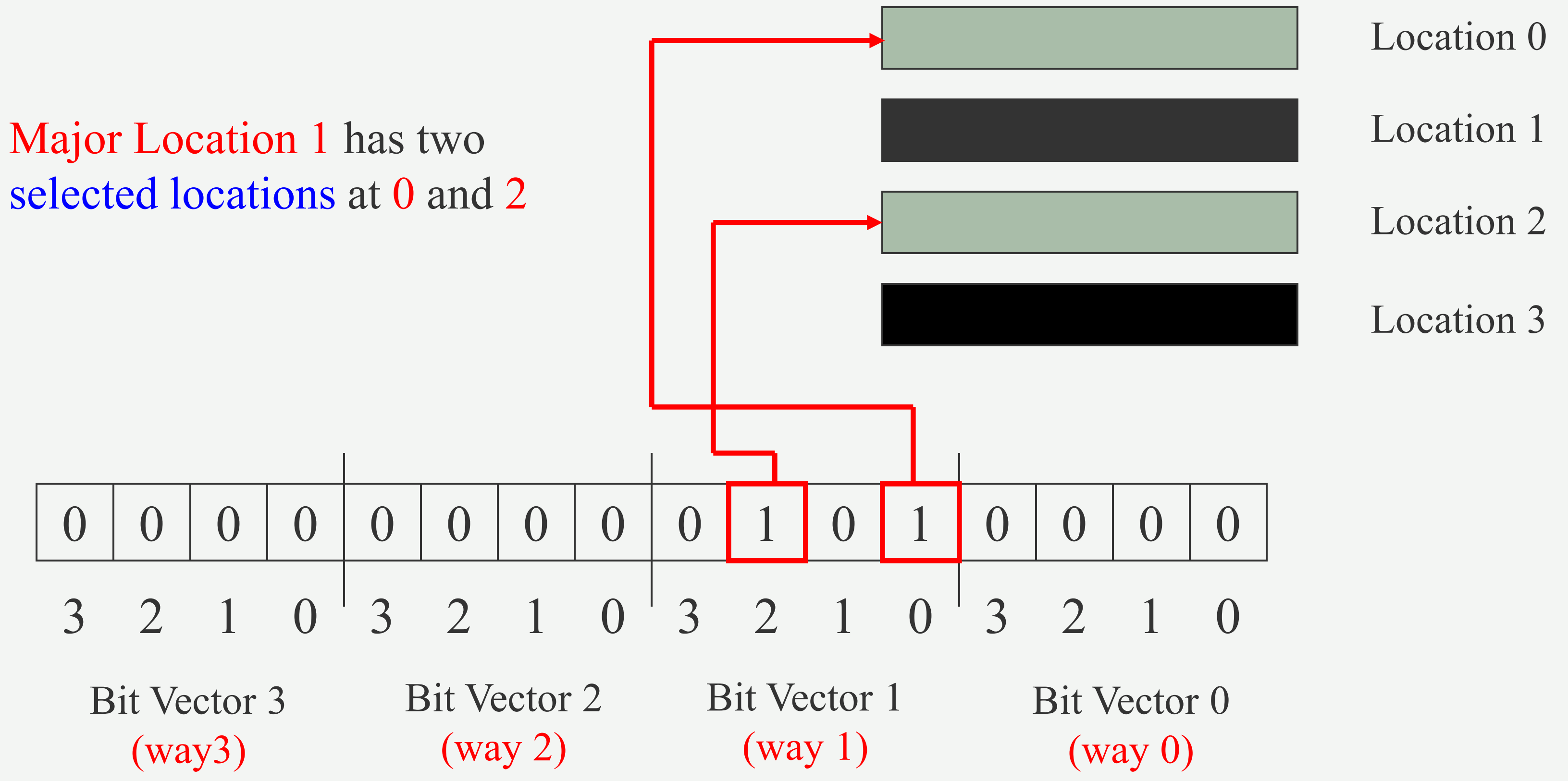
- Major location contains a most recent used (MRU) block either loaded from memory or just accessed.

Multi-column Caches: Selected Locations

- Multiple blocks can be direct-mapped to the same major location, but only MRU is the major.
- The non-MRU blocks are stored in other empty locations in the set: Selected Locations.
- If ``other locations" are used for their own major locations, there will be no space for selected ones.
- Swap:
  - A block in selected location is swapped to major location as it becomes MRU.
  - A block in major location is swapped to a selected location after a new block is loaded in it from memory.

Multi-Column: Indexing Selected Locations

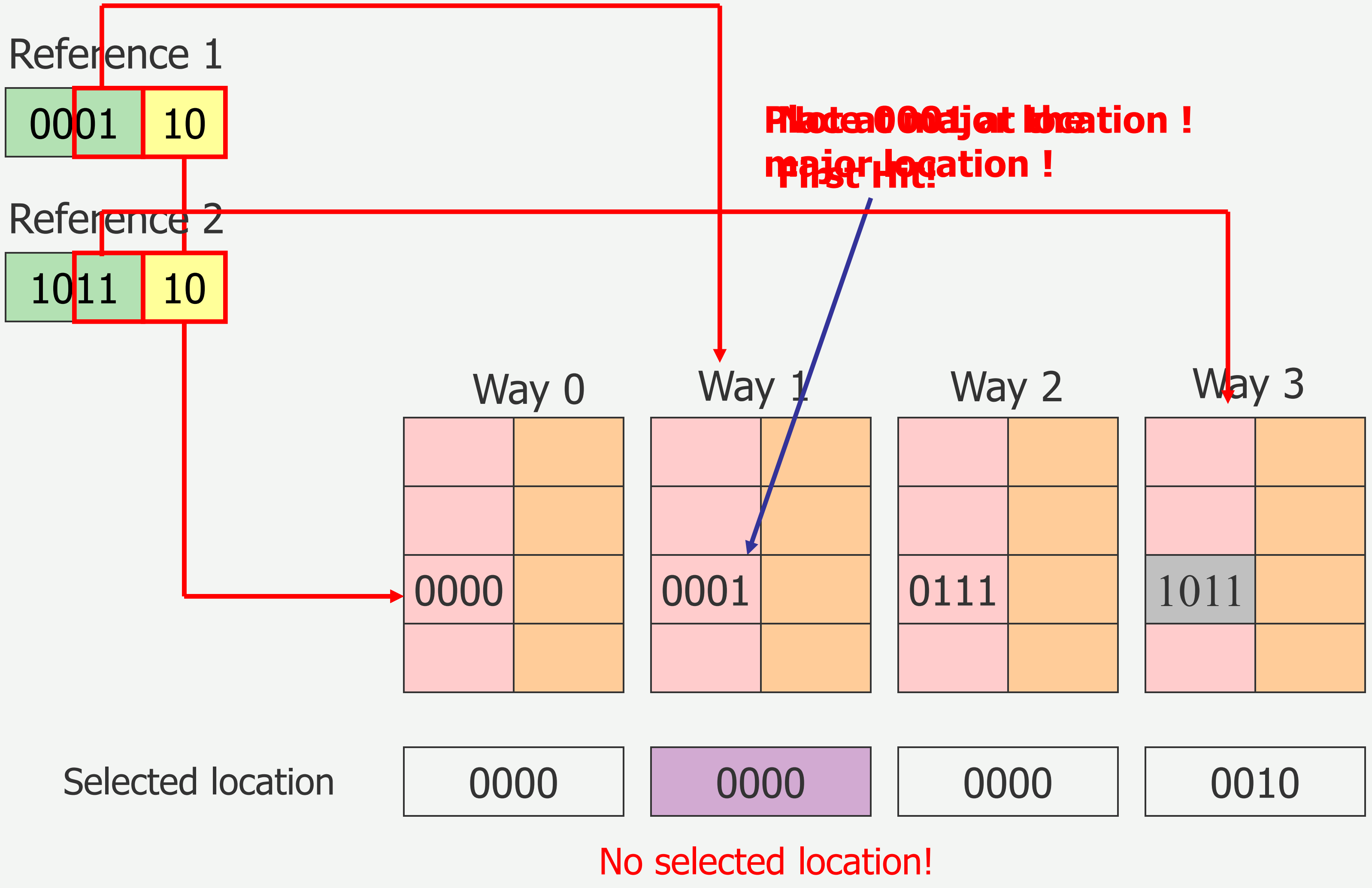
- The selected locations associated with its major location are indexed for a fast search.



Summary of Multi-column Caches

- A major location in a set is the direct-mapped location of MRU.
- A selected location is the direct-mapped location but non-MRU.
- A selected location index is maintained for each major location.
- A ``swap” is used to ensure the block in the major location is always MRU.

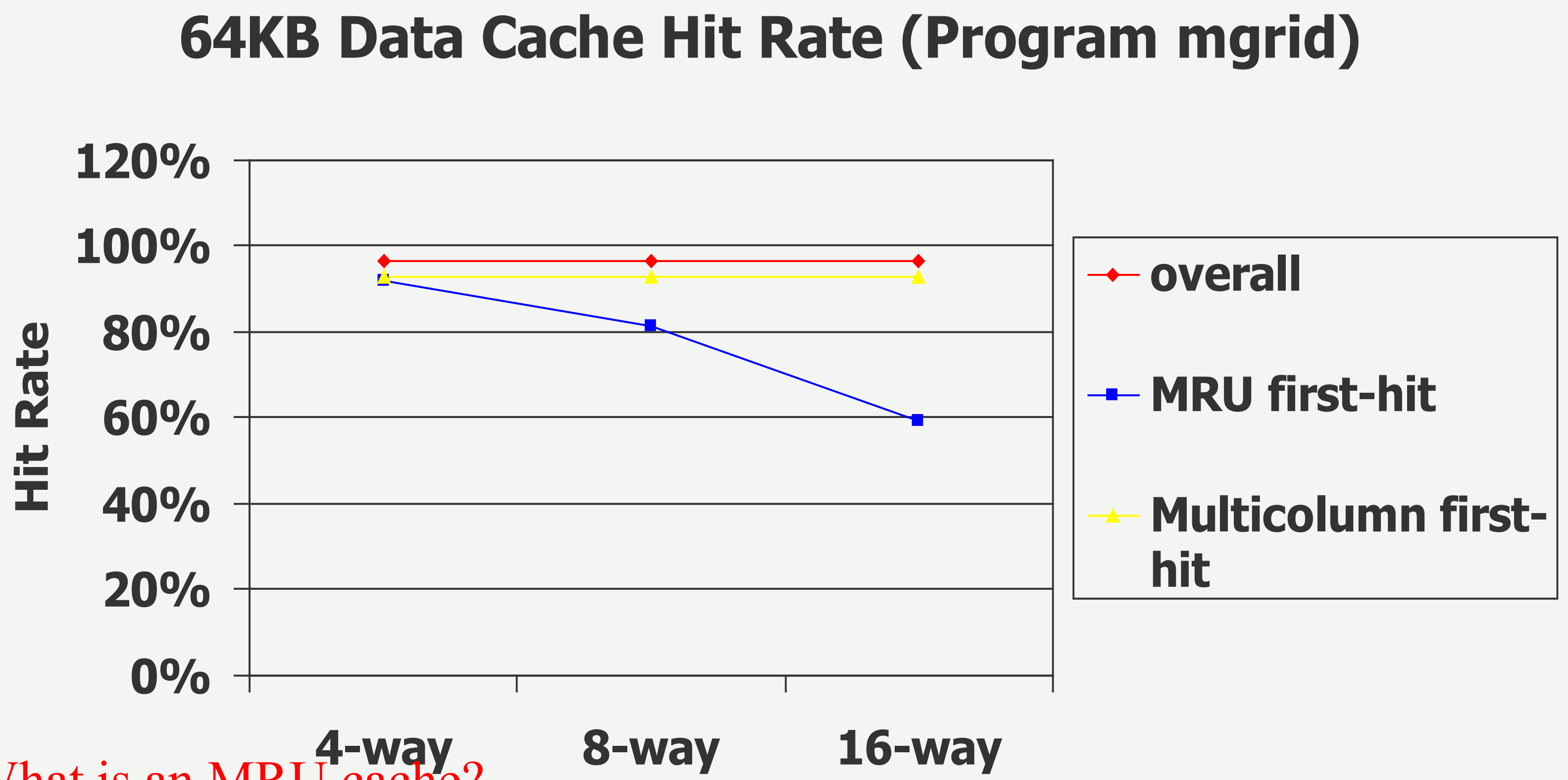
Multi-column Cache Operations



Performance Summary of Multi-column Caches

- Hit ratio to the major locations is about 90%.
- The hit ratio is higher than that of direct-mapped cache due to high associativity while keeps low access time of direct-mapped cache in average.
  - First-hit is equivalent to direct-mapped.
  - Non-first hits are faster than set-associative caches.
- Outperforming set associative caches

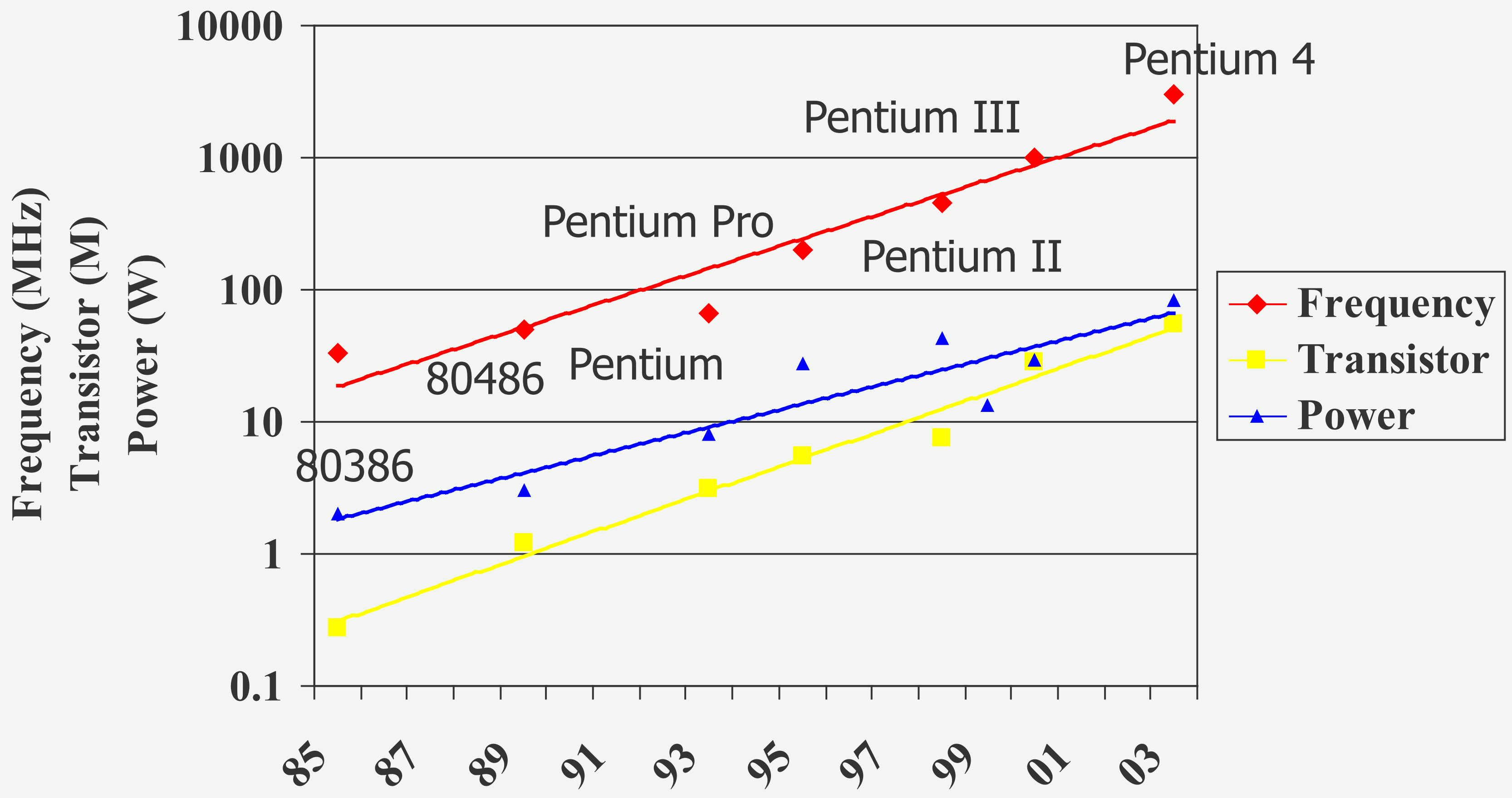
Comparing First-hit Ratios between Multicolumn and MRU Caches



Note: What is an MRU cache?

An MRU bit map is used for a set associative cache, (1 bit for way, 2 bits for 4-way, …), which remembers the most recent used way as the prediction of the next access. This bit map is used for the way prediction.

Multi-column Technique is Critical for Low Power Caches by targeting specific ways



Source: Intel.com

The impact of multicolumn caches

- The concept of multicolumn caches has influenced the R&D in the set-associative cache products:
  - A tag-less cache
  - K-way direct mapped cache
  - The direct-to-data cache
  - Highly associative caches for low power processors
  - A way memorization cache
  - Way selection based on tag bits
  - In ARM Cortex R chip, cache mapping “directly access one of the ways in a set”