Chapter 09: Caches

Lesson 03:

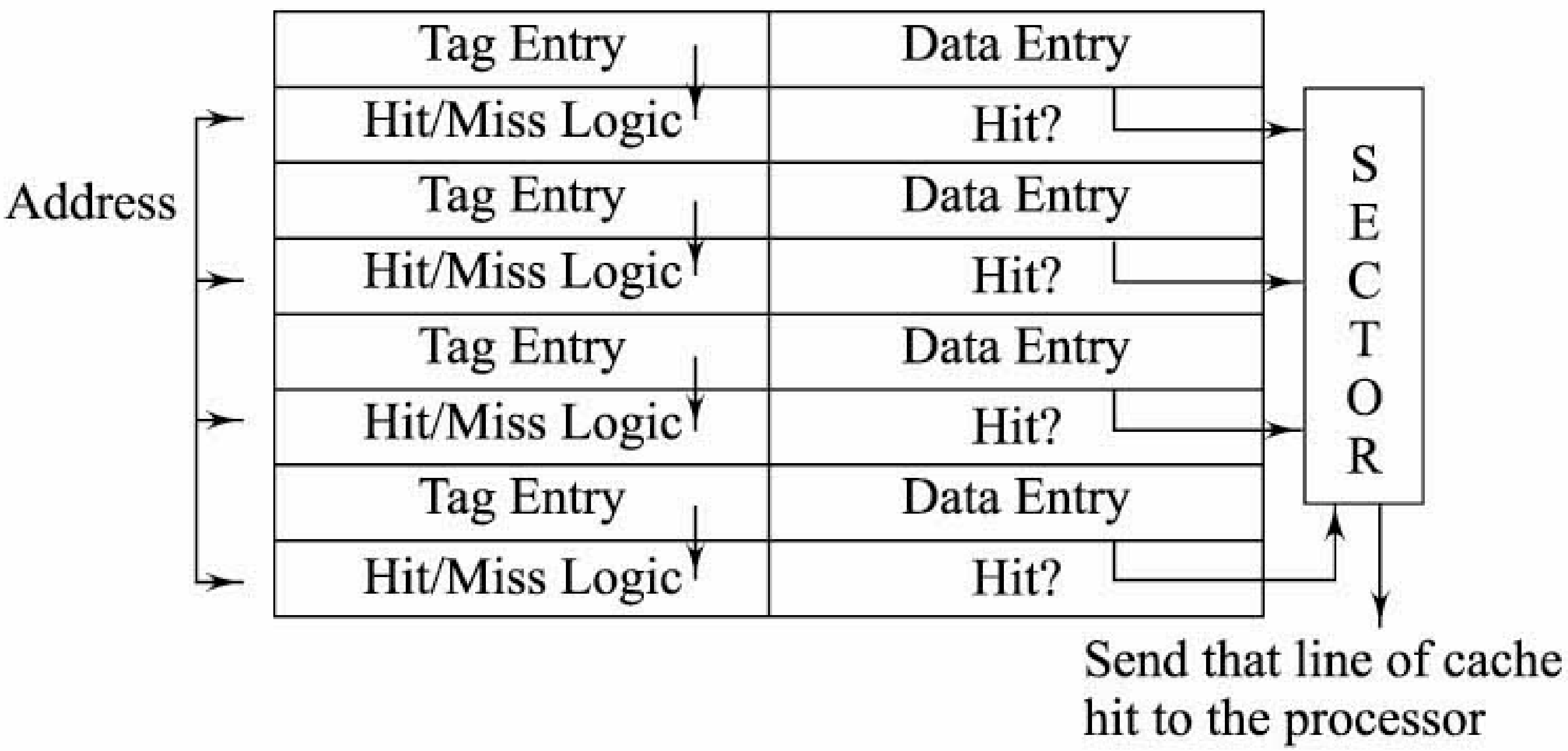
Cache Associativity

Objective

- Understand fully Associative Cache– where any address can be stored in any line in the cache
    - Learn Direct Mapped Cache─ opposite stream, each memory address can only be stored in one location in the cache
    - Learn Set Associative Cache ─ a compromise between direct and fully associative, a fixed number of locations (called a set) that a given address may be stored in
    - Learn Two or higher way Set Associative Cache

Associative Cache Memory

Cache associative memory



Address

Tag Entry	Data Entry
Hit/Miss Logic	Hit?
Tag Entry	Data Entry
Hit/Miss Logic	Hit?
Tag Entry	Data Entry
Hit/Miss Logic	Hit?
Tag Entry	Data Entry
Hit/Miss Logic	Hit?

SECTOR

Send that line of cache hit to the processor

Cache Fully Associative Memory

Full Associativity

- Allow any address to be stored in any line in the cache
- When a memory operation is sent to the cache, the address of the request must be compared to each entry in the tag array to determine whether the data referenced by the operation is contained in the cache

Full Associativity

- Fully associative caches are generally still implemented with separate tag and data arrays

Direct Mapped cache

Direct Mapped Caches

- Direct-mapped caches are the opposite extreme from fully associative caches
- In a direct-mapped cache, each memory address can only be stored in one location in the cache

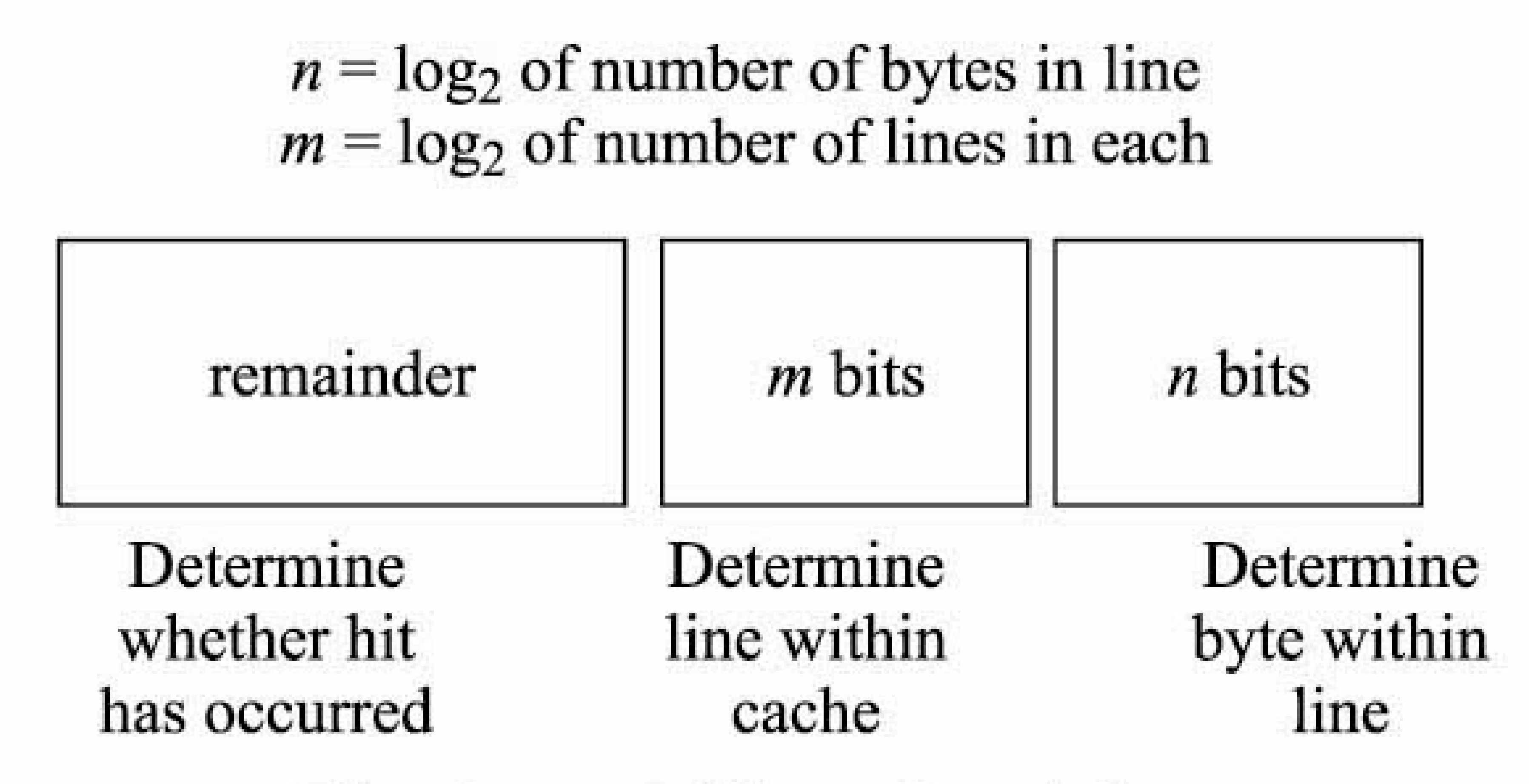
Direct mapped Cache

- When a memory operation is sent to a direct-mapped cache, a subset of the bits in the address is used to select the line in the cache that may contain the address
- Another subset of the bits is used to select the byte within a cache line that the address points to

Direct mapped Cache

- In general, the *n* lowest-order bits in the address are used to determine the position of the address within its cache where *n* is the base-2 logarithm of the number of bytes in the line
- The *m* next higher-order bits, where *m* is the base-2 logarithm of the number of lines in the cache, are used to select the line in which the address may be stored

Address lower and higher order bits breakdown



$n = \log_2$ of number of bytes in line

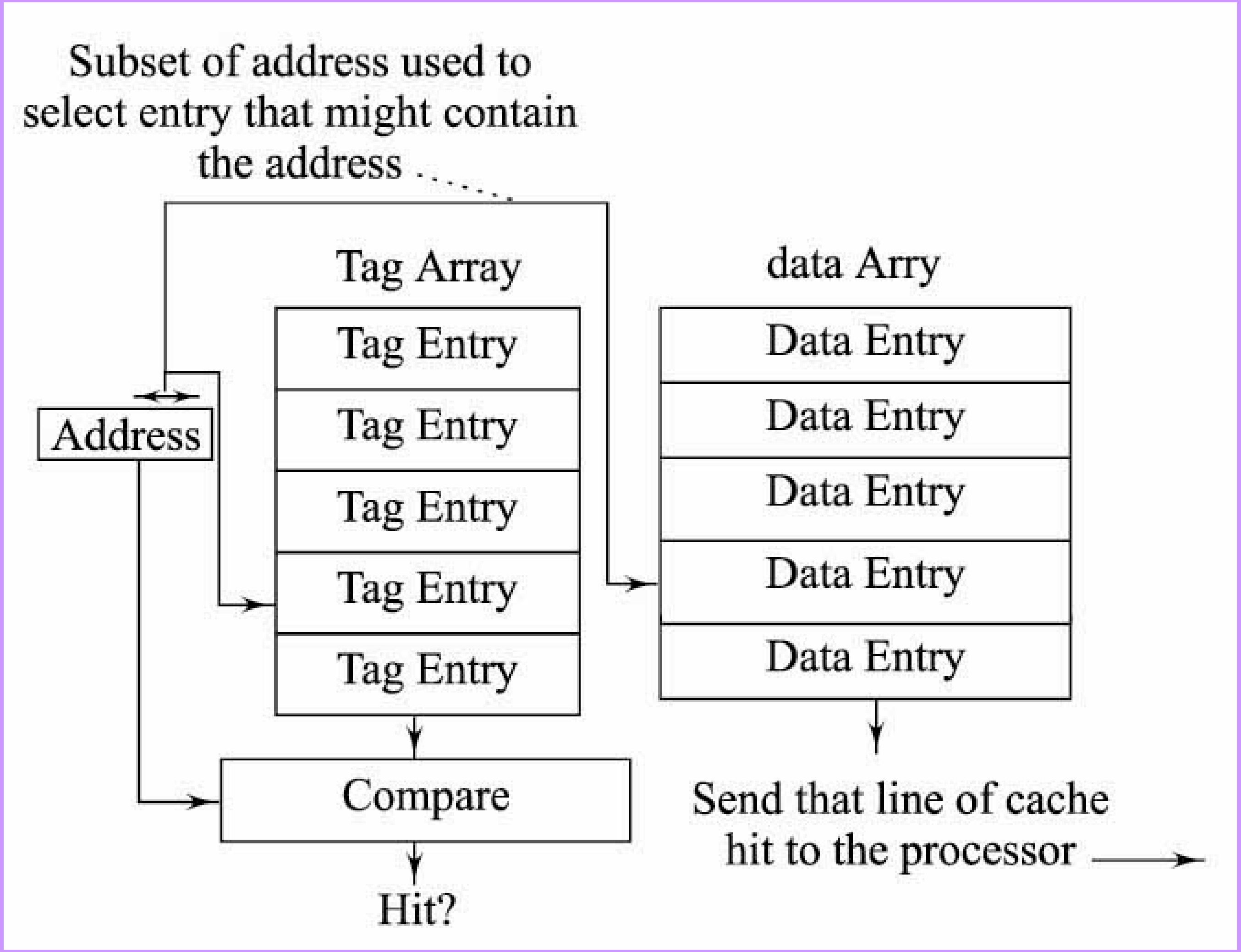
$m = \log_2$ of number of lines in each

remainder — Determine whether hit has occurred

m bits — Determine line within cache

n bits — Determine byte within line

Direct mapped cache memory



Direct mapped Cache significantly less chip space

- Direct-mapped caches have the advantage of requiring significantly less chip space to implement than fully associative caches because they only require one comparator to determine if a hit has occurred, while fully associative caches require one comparator for each line in the cache

Direct mapped Cache lower Access Time

- In addition, direct-mapped caches usually have lower access times because there is only one comparison to examine to determine if a hit has occurred, while a fully associative cache must examine each of the comparisons and select the appropriate word of data to send to the processor

Direct mapped cache lower Hit Rate

- Direct-mapped caches tend to have lower hit rates than fully associative caches, due to conflicts between lines that map into the same space in the cache
- Each address can only be placed in one location in the cache, which is determined by the *m* address bits

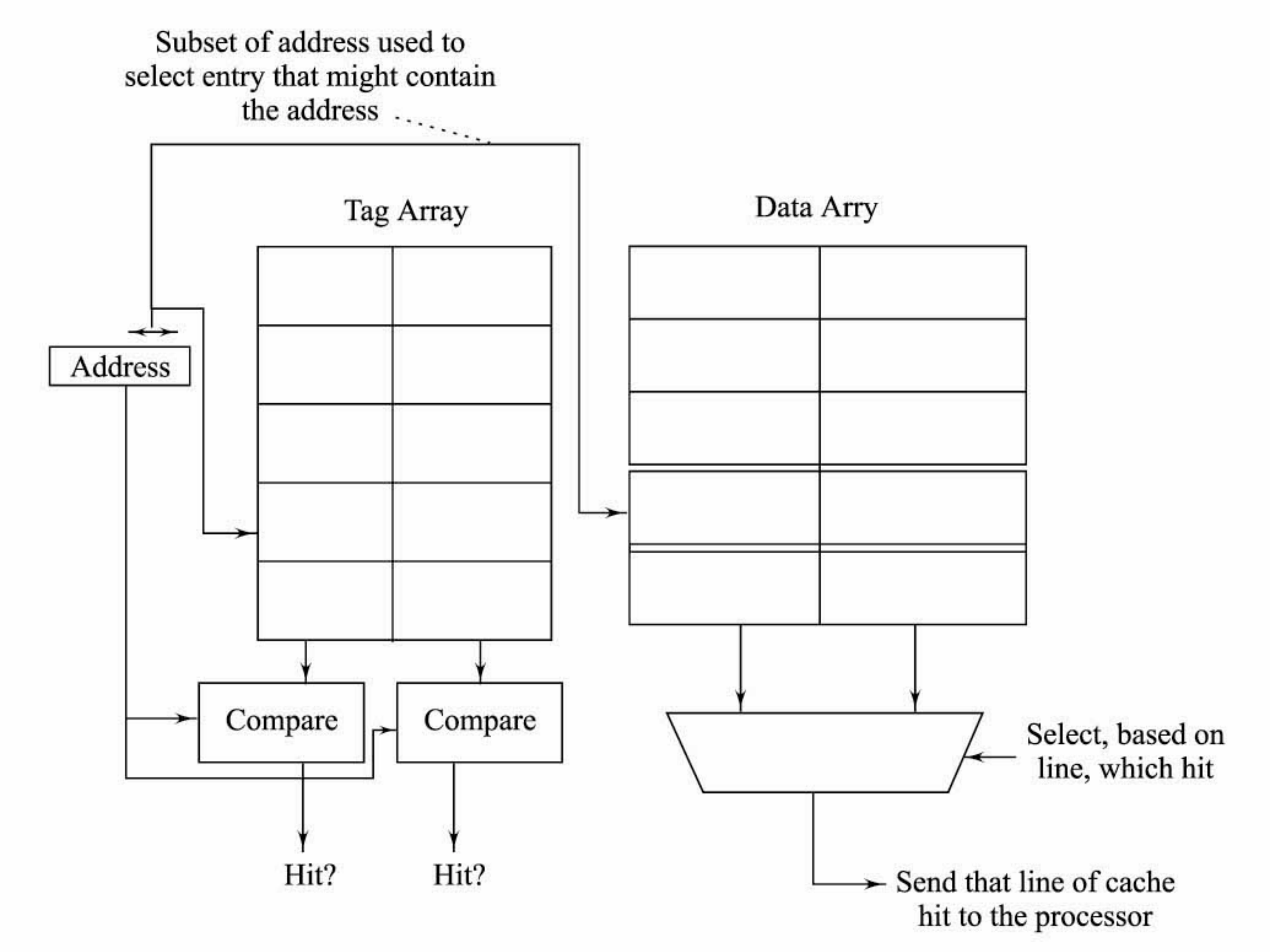
Set Associative Cache

Set Associative Cache

- Set associative caches are a compromise between fully associative caches and direct-mapped caches. In a set associative cache, there are a fixed number of locations (called a set) that a given address may be stored in. The number of locations in each set is the associative of the cache.

Two Way Set Associative Cache Block

Two-way set associative cache memory


Subset of address used to select entry that might contain the address
Tag Array
Data Arry
Address
Compare
Compare
Hit?
Hit?
Select, based on line, which hit
Send that line of cache hit to the processor

Two-Way Set Associative

- Like a direct mapped cache, a subset of the address bits is used to select the set that might contain the address
- There are two tags that must be compared with the address of the memory reference to determine if a hit has occurred
- If either of the tag matches the address, a hit has occurred and the corresponding line of the data array is selected

Higher than two Way Set Associative Cache Block

Higher Associativity

- More comparators to determine if a hit has occurred
- A set-associative cache will have fewer sets than a direct-mapped cache that contains the same number of lines and will therefore use fewer bits of an address to select the set that the address will be stored in

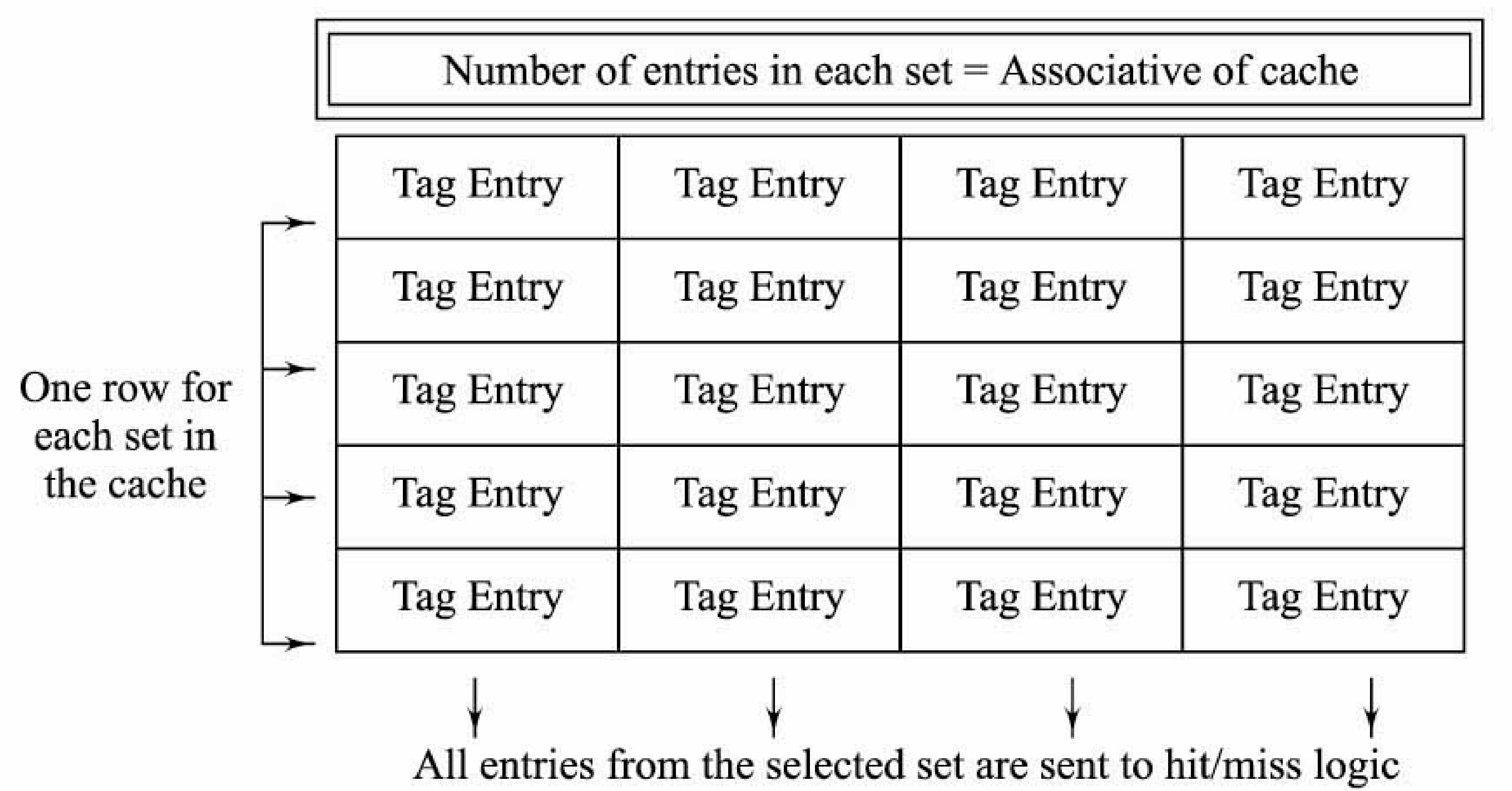
Higher Associativity

- Computing the number of lines in the cache and dividing by associativity can find the number of sets in a cache

Four-Way Set Associative

- There are four tags that must be compared with the address of the memory reference to determine if a hit has occurred
- If either of the tag matches the address, a hit has occurred and the corresponding line of the data array is selected

4-way set associative cache



Number of entries in each set = Associative of cache

Tag Entry	Tag Entry	Tag Entry	Tag Entry
Tag Entry	Tag Entry	Tag Entry	Tag Entry
Tag Entry	Tag Entry	Tag Entry	Tag Entry
Tag Entry	Tag Entry	Tag Entry	Tag Entry
Tag Entry	Tag Entry	Tag Entry	Tag Entry

One row for each set in the cache

All entries from the selected set are sent to hit/miss logic

Example

- If a cache has a capacity of 16 kB and a line length of 128 bytes, find how many sets does the cache have if it is 2-way, 4-way, or 8-way set-associative

Solution

- With 128-byte lines, the cache contains a 16 kB ÷ 128 = 16 × 1024 / 128 = a line length of 128 bytes in 128 lines
- The number of sets in the cache = the number of lines divided by the associativity
- The cache has 64 sets = 128/2 if it is 2-wav set-associative
- 32 sets = 128/4 if 4-way set-associative
- 16 sets = 128/16 if 8-way set-associative

Example

- Find how many sets are there in a two-way set-associative cache with 32 kB capacity and 64-byte lines
- Find how many bits of the address are used to select a set in this cache
- Find bits of the address in an eight way set-associative cache with the same capacity and line-length

Solution

- A 32 kB cache with 64 byte lines contains = 32 × 1024 ÷ 64 = 32 × 16 = 512 lines of data
- In a two-way set-associative cache, each set contains 2 lines
- 512 ÷ 2= 256 sets in the cache
- $\text{Log}_2(256) = 8$, so 8-bits of an address are used to select a set that the address maps to

Solution

- In a 8-way set-associative cache, each set contains 8 lines
- 512 ÷ 8= 64 sets in the cache
- $\text{Log}_2(64) = 6$, so 6-bits of an address are used to select a set that the address maps to

Better Hit Rates than direct mapped caches

- But worse hit rates than fully associative caches of the same size
- Because allowing each address to be stored in multiple locations eliminates some, but not all, of the conflicts between cache lines that occur in a direct mapped cache

Relationship with capacity

Relationship with capacity

- The difference in hit rate is a function of the capacity of cache, the degree of associativity, and the data referenced by a given program
- Some programs reference large blocks of contiguous data, leading to few conflicts, while others reference many disjoint data objects, which can lead to conflicts if the objects map to the same sets in the cache

Larger Caches

- The larger a cache is, the less benefit it tends to see from associativity, since there is a lower probability that any two addresses will map onto the same space in the cache
- Finally, successive increases in associativity have diminishing returns

Larger Caches

- Going from a direct-mapped cache to a two-way set-associative cache usually causes significant reductions in the miss rate
- Increasing to four-way set-associative (associativity is usually a power of 2 to simplify the hardware, but other associativities are possible) has a less significant effect, and increasing beyond that tends to have very little effect, except for extremely small caches

Associativity and Miss Rate

Associativity and Miss Rate

- Going from a direct-mapped cache to a two-way set-associative cache usually causes significant reductions in the miss rate
- Increasing to four-way set-associative (associativity is usually a power of 2 to simplify the hardware, but other associativities are possible) has a less significant effect, and increasing beyond that tends to have very little effect, except for extremely small caches.

Common Caches

- Two-way and four-way set-associative caches most common in current microprocessors

Summary

We Learnt

- Fully Associative cache having comparator in each Line
- Direct Address Mapped Cache one comparator for the Cache
- Set associative cache each set having a comparator
- Two-way Set associative has two comparators, 4-way four, ..

End of Lesson 03:

Cache Associativity