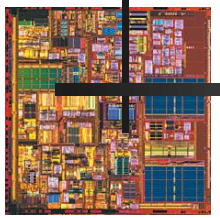
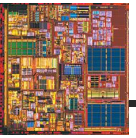
ECE 411
Fall 2015

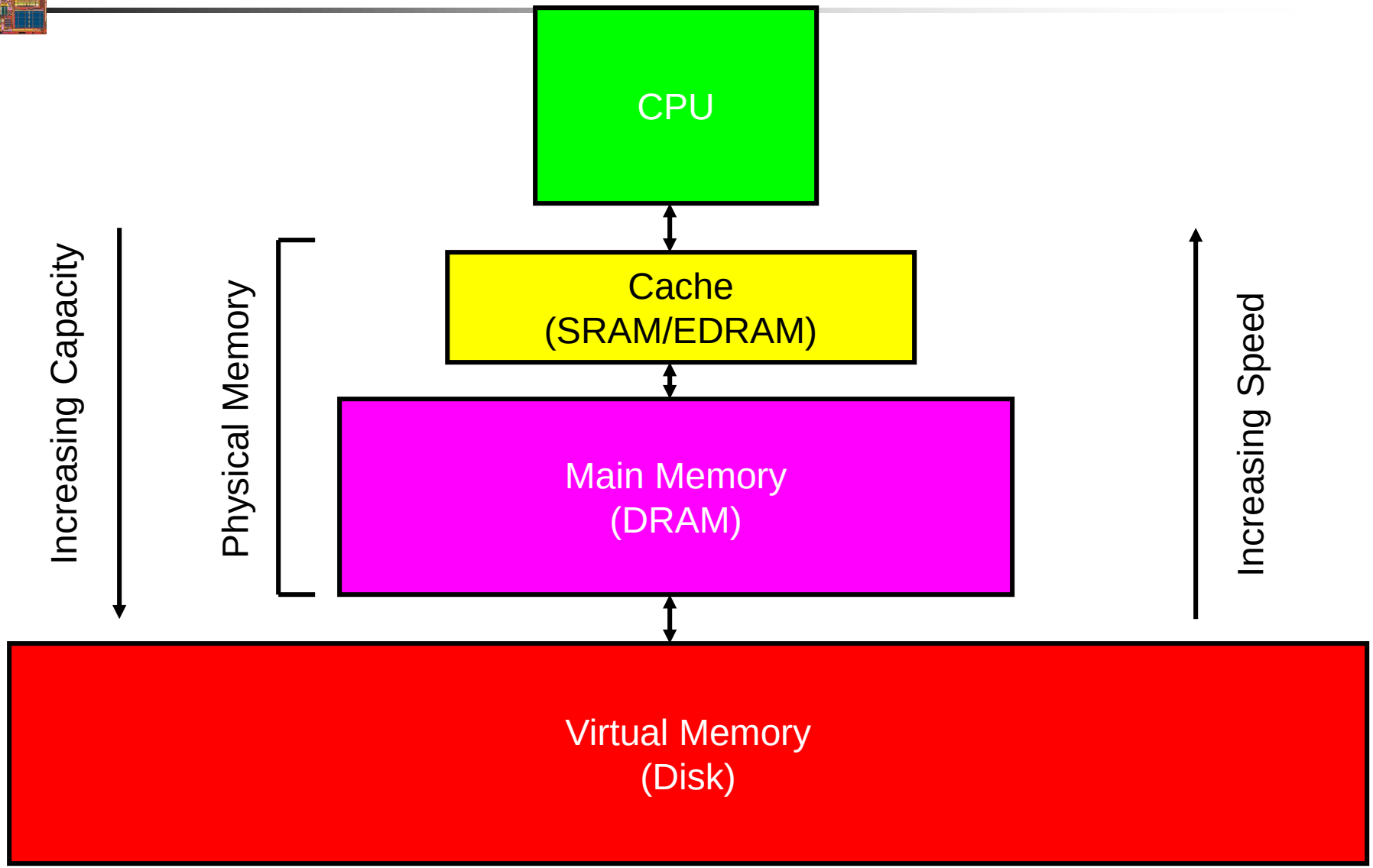


Lecture 5

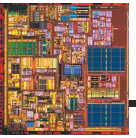
Cache Memories

ILLINOIS




CPU
Cache
(SRAM/EDRAM)
Main Memory
(DRAM)
Virtual Memory
(Disk)
Increasing Capacity
Physical Memory
Increasing Speed

Performance of a 2-level Memory Hierarchies



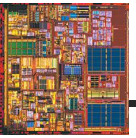
Basic Formula:

- $T_{avg} = P_{hit} * T_{hit} + P_{miss} * T_{miss}$

T_{hit} = time to complete the memory reference if we hit in the higher level of the hierarchy

T_{miss} = time to complete the memory reference if we miss and have to go down to the next level

P_{hit}, P_{miss} = Probabilities of hitting or missing in the level

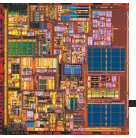


Example 1

A memory system consists of a cache and a main memory. If it takes 1 cycle to complete a cache hit, and 100 cycles to complete a cache miss, what is the average memory access time if the hit rate in the cache is 97%?

$$T_{avg} = 1 * 0.97 + 100 * 0.03 = 3.97$$

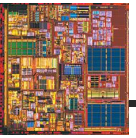
Example 2



- A memory system has a cache, a main memory, and a disk virtual memory. If the hit rate in the cache is 98% and the hit rate in the main memory is 99% (among the accesses that miss from cache), what is the average memory access time if it takes 2 cycles to access the cache, 150 cycles to complete a miss from cache but hit in the main memory, and 100,000 cycles to complete a miss from both the cache and the main memory?

$$T_{avg} = 2 * 0.98 + 150 * 0.02*0.99 + 100{,}000*0.02*0.01$$
$$= 1.96 + 2.97 + 20 = 24.93$$

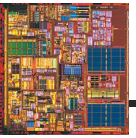
ILLINOIS



Four Central Questions in Designing a Cache

- P-I-R-W:
  - Placement
    - Where can a block of memory go?
  - Identification
    - How do I find a block of memory?
  - Replacement
    - How do I make space for new blocks?
  - Write Policy
    - How do I propagate changes?
- Need to consider these for all levels of the memory hierarchy
  - L1/L2/L3 caches now
- Main memory, disks have similar issues, addressed later

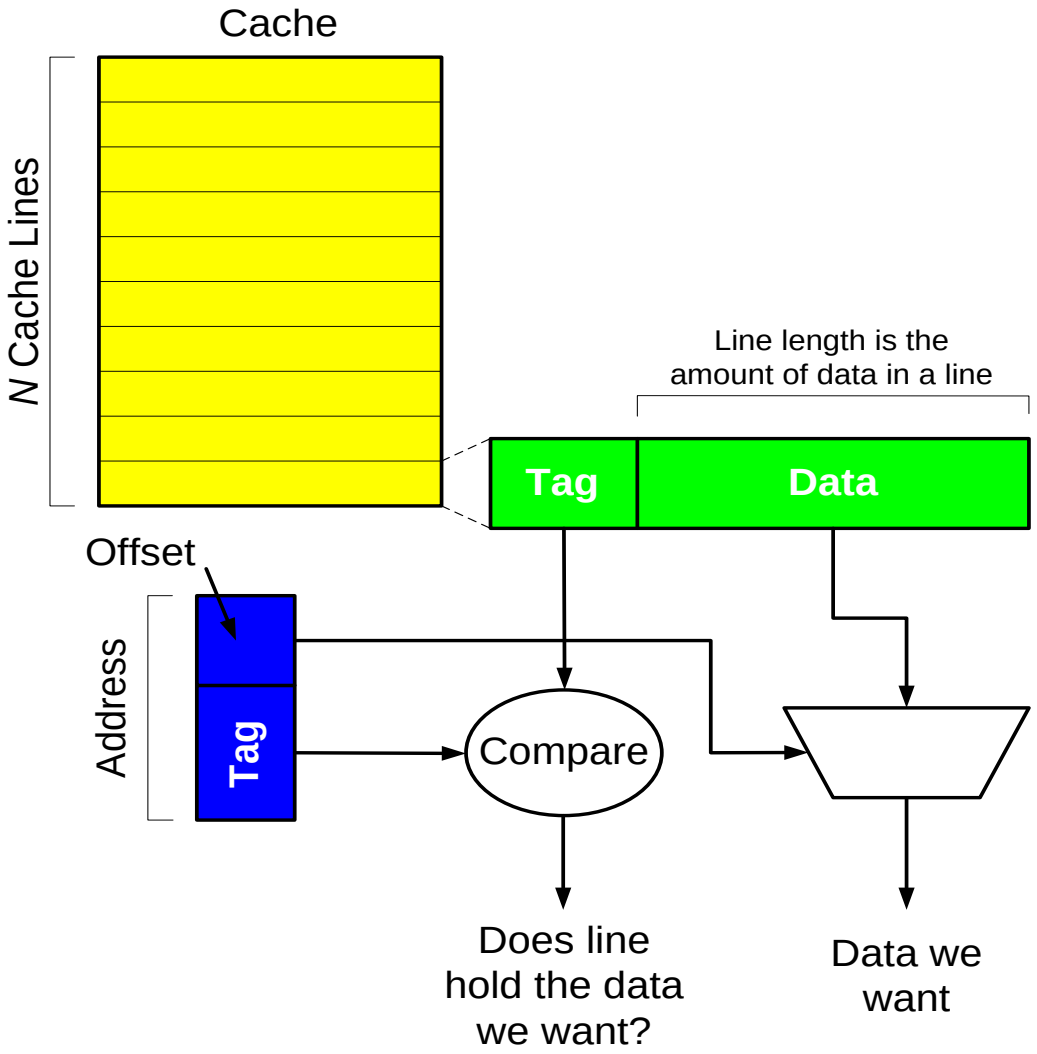
Describing Caches



We characterize a cache using 7 parameters

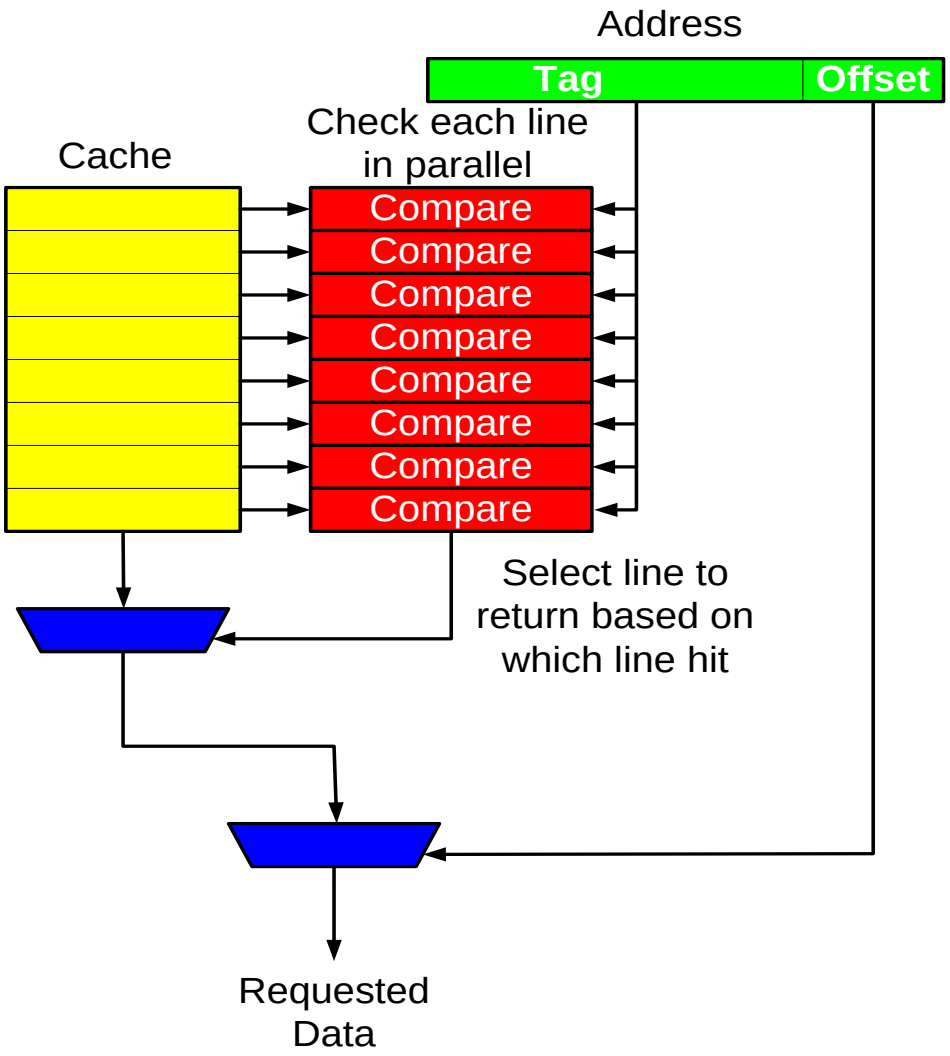
- Access Time: T_{hit}
- Capacity: the total amount of data the cache can hold
  - # of blocks * block size
- Block (line) Size: The amount of data that gets moved into or out of the cache as a chunk
  - Analogous to page size in virtual memory
- What happens on a write?
- Replacement Policy: What data is replaced on a miss?
- Associativity: How many locations in the cache is a given address eligible to be placed in?
- Unified, Instruction, Data: What type of data is kept in the cache?
  - We'll cover this in more detail later

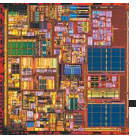
Cache Operation – Assuming one-level cache



- On memory access, look in the cache first
- If the address we want is in the cache, complete the operation, usually in one or two cycles
- If not, complete the operation using the main memory (many cycles)

Fully-Associative: Anything Can Go Anywhere





Simple Fully Associative cache in action

Sequence of memory references in byte addresses:

24, 20, 04, 12, 20, 44, 04, 24, 44

Assuming 4 bytes blocks, block addresses:

6, 5, 1, 3, 5, 11, 1, 6, 11

Assume 4 blocks

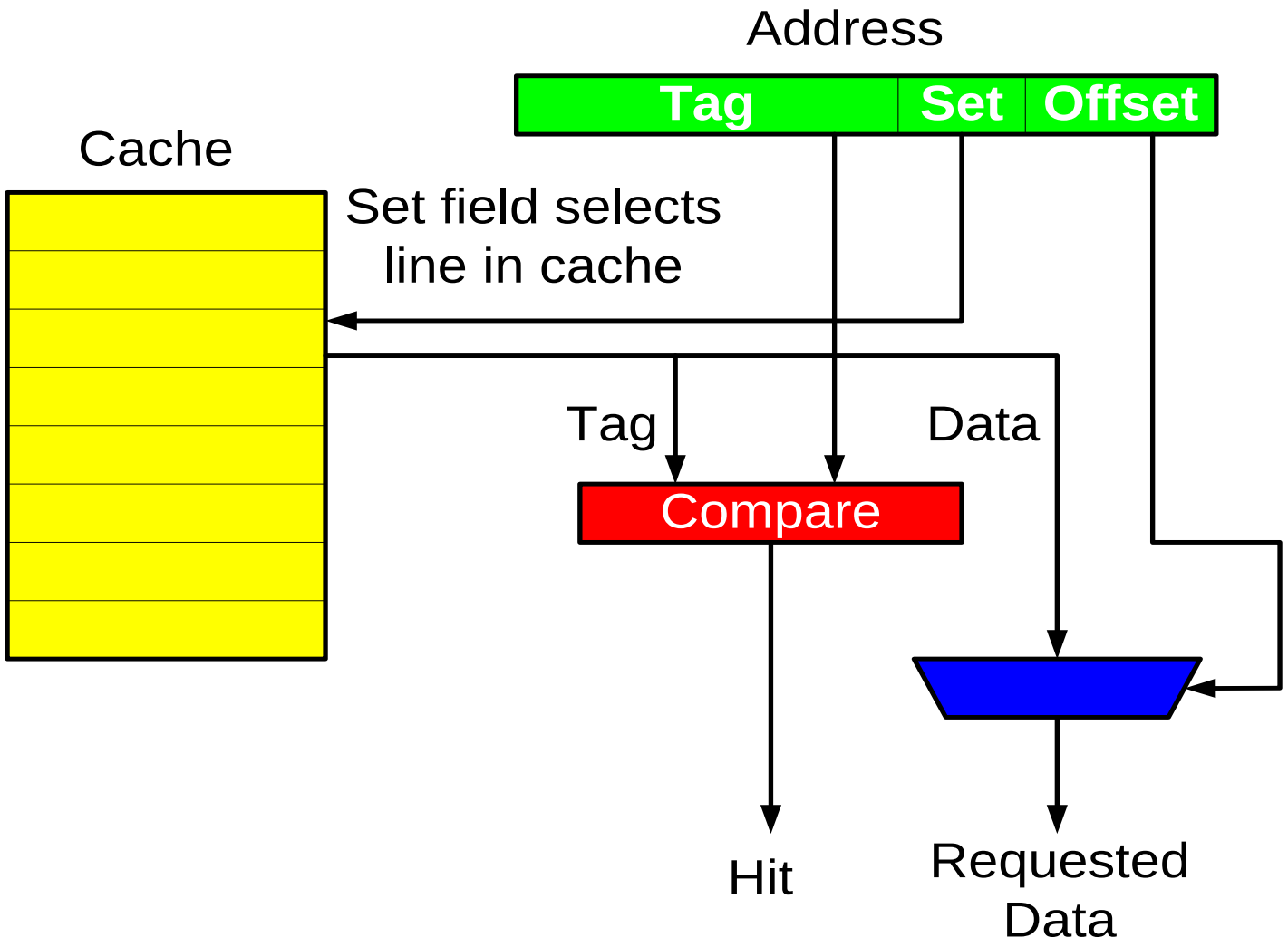
Start with an empty cache, assuming a least recently used replacement

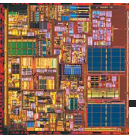
M, M, M, M, H, M (replace 6), H, M (replace 3), H

Use a stack to keep track of recency of access to each block

ILLINOIS

Direct-Mapped:
One cache location for each address





Simple Direct Mapped Cache in Action

Sequence of memory references in byte addresses:

24, 20, 04, 12, 20, 44, 04, 24, 44

Assuming 4 bytes blocks, block addresses:

6, 5, 1, 3, 5, 11, 1, 6, 11

Assume 4 blocks,

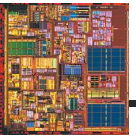
Start with an empty cache

2, 1, 1, 3, 1, 3, 1, 2, 3

M, M, M, M, M, M, M H, H

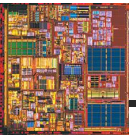
ILLINOIS

Direct-Mapped vs. Fully-Associative



- Direct-Mapped
  - Require less area
    - Only one comparator
    - Fewer tag bits required
  - Fast: can return data to processor in parallel with determining if a hit has occurred
  - Conflict misses reduce hit rate
- Fully-Associative
  - No conflict misses, therefore higher hit rate in general
  - Need one comparator for each line in the cache
- Design trade-offs
  - For a given chip area, will you get a better hit rate with a fully-associative cache or a direct-mapped cache with a higher capacity?
  - Do you need the lower access time of a direct-mapped cache?

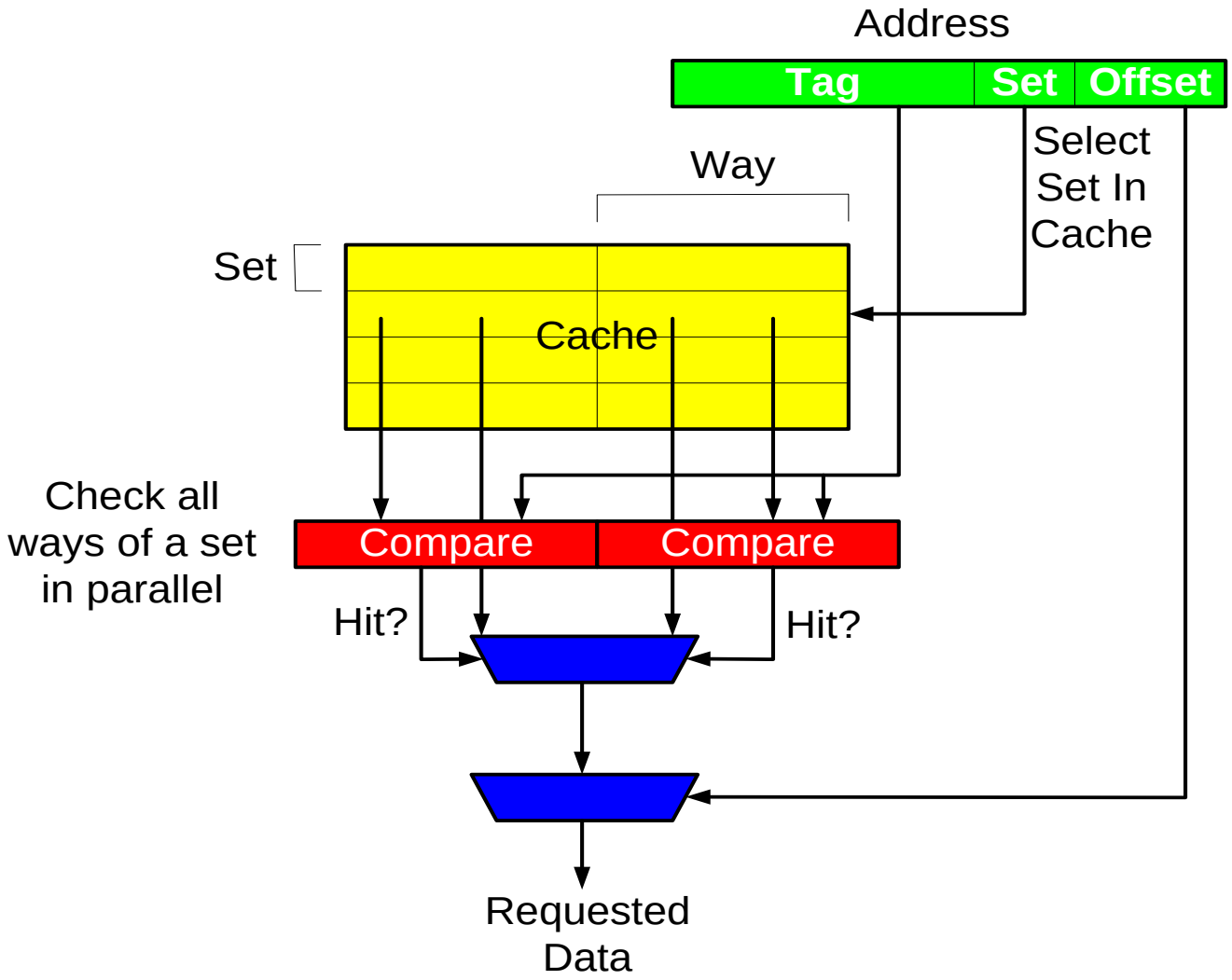
Talking About Cache Misses

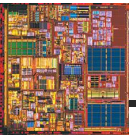


In single-processor systems, cache misses can be divided into three categories:

- Compulsory: Misses caused by the first reference to each line of data
  - In an infinitely-large fully-associative cache, these would be the only misses
- Capacity: Misses caused because a program references more data than will fit in the cache
- Conflict: Misses caused because more lines try to share a specific place in the cache than will fit

Compromise: Set-Associative Caches





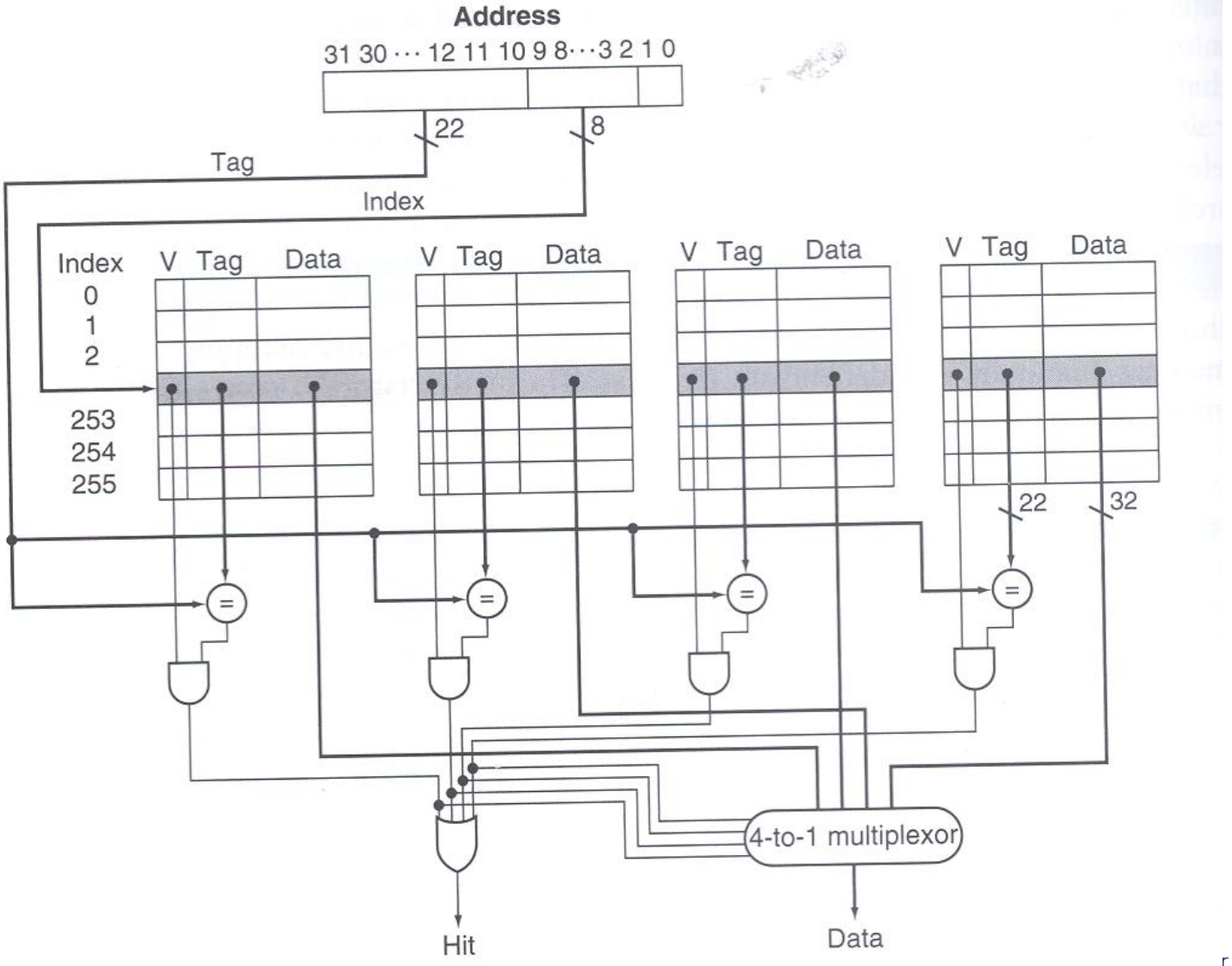
2-way set associative cache in action

(Block number) modulo (Number of sets in the cache)

Sequence of memory references: 22, 26, 22, 26, 16, 3, 16, 18, 16

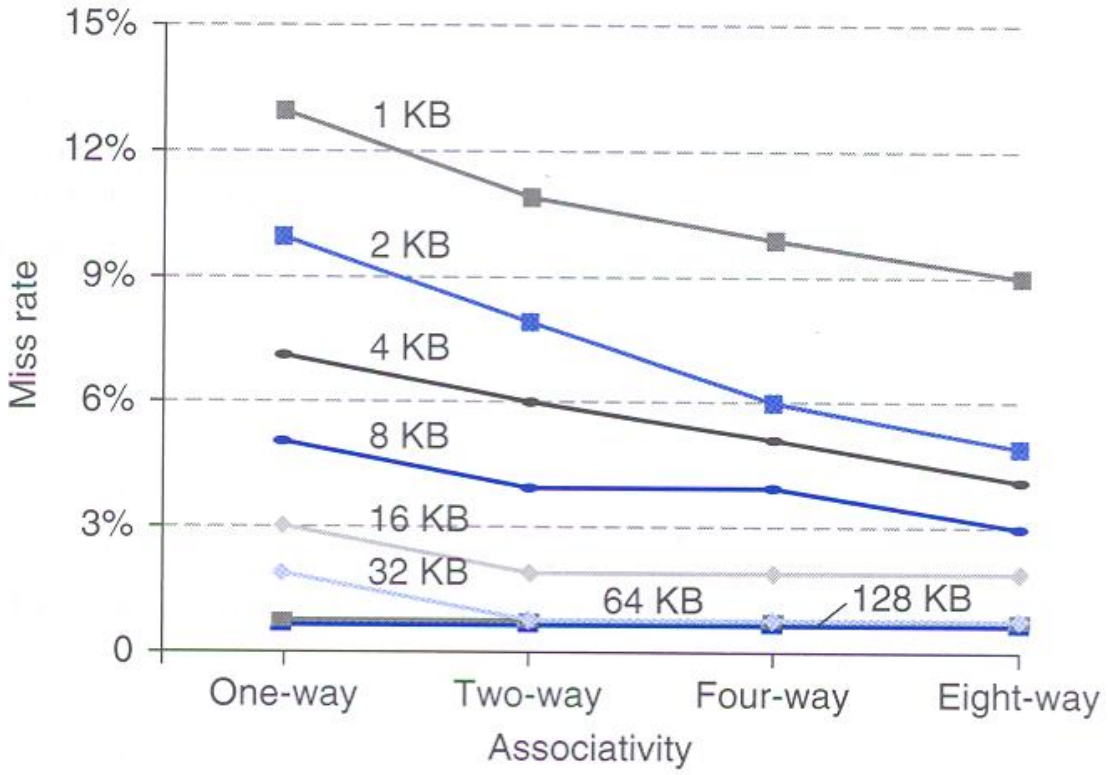
ILLINOIS

A 4-way set associative cache



ILLINOIS

Data cache miss rates

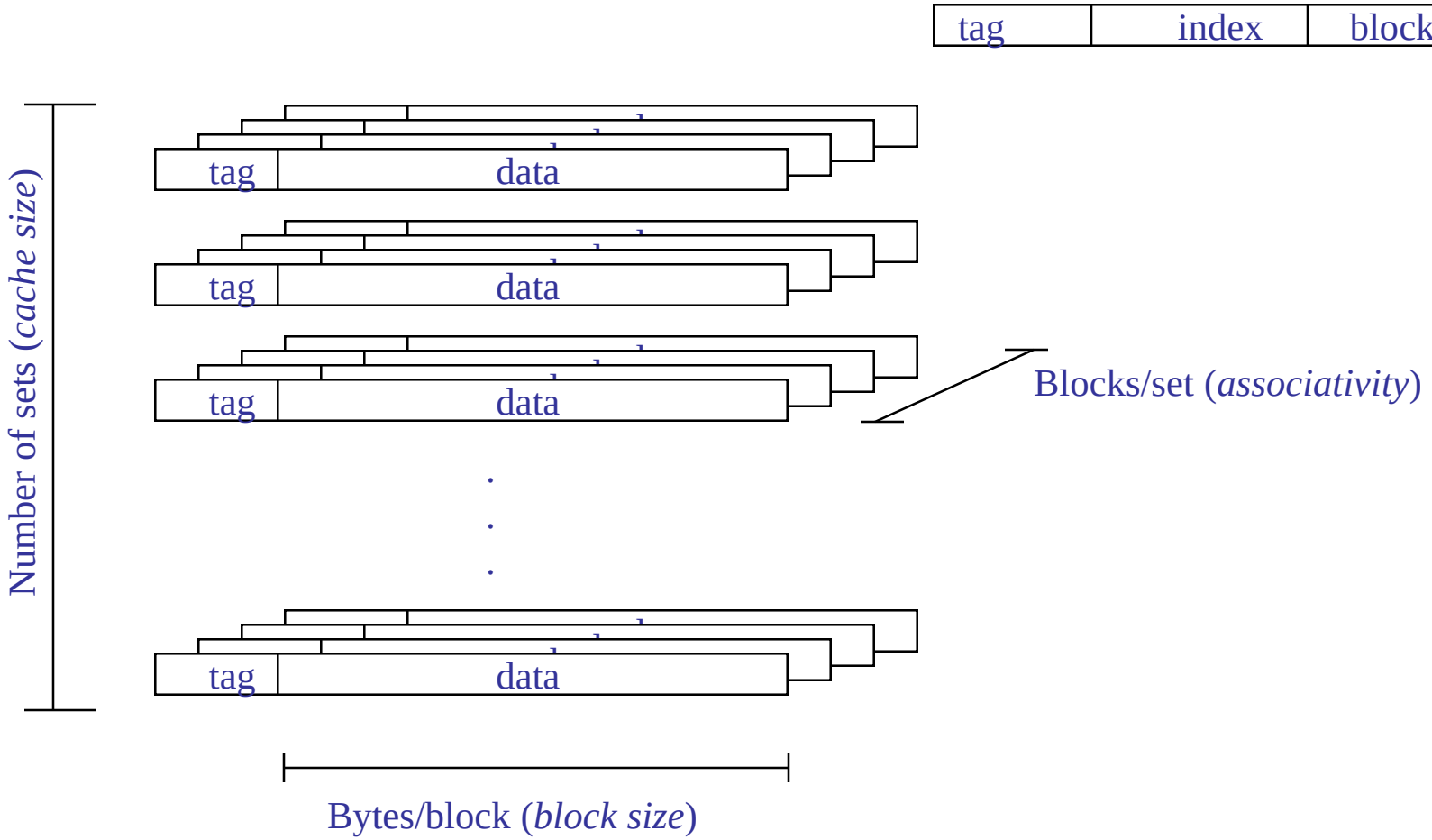


- Benefits from 1 way to 2 way is larger
- Smaller caches obtain more benefit from associativity because the base miss rate of a smaller cache is larger

ILLINOIS

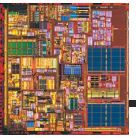
Cache Organization -- Recap

- A typical cache has three dimensions



ILLINOIS

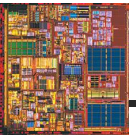
Cache Parameters



Cache size = Number of sets * block size * associativity

- 128 blocks, 32-byte blocks, direct mapped, size = ?
- 128 KB cache, 64-byte blocks, 512 sets, associativity = ?

ILLINOIS



Choosing bits for the index

Anatomy of an address:

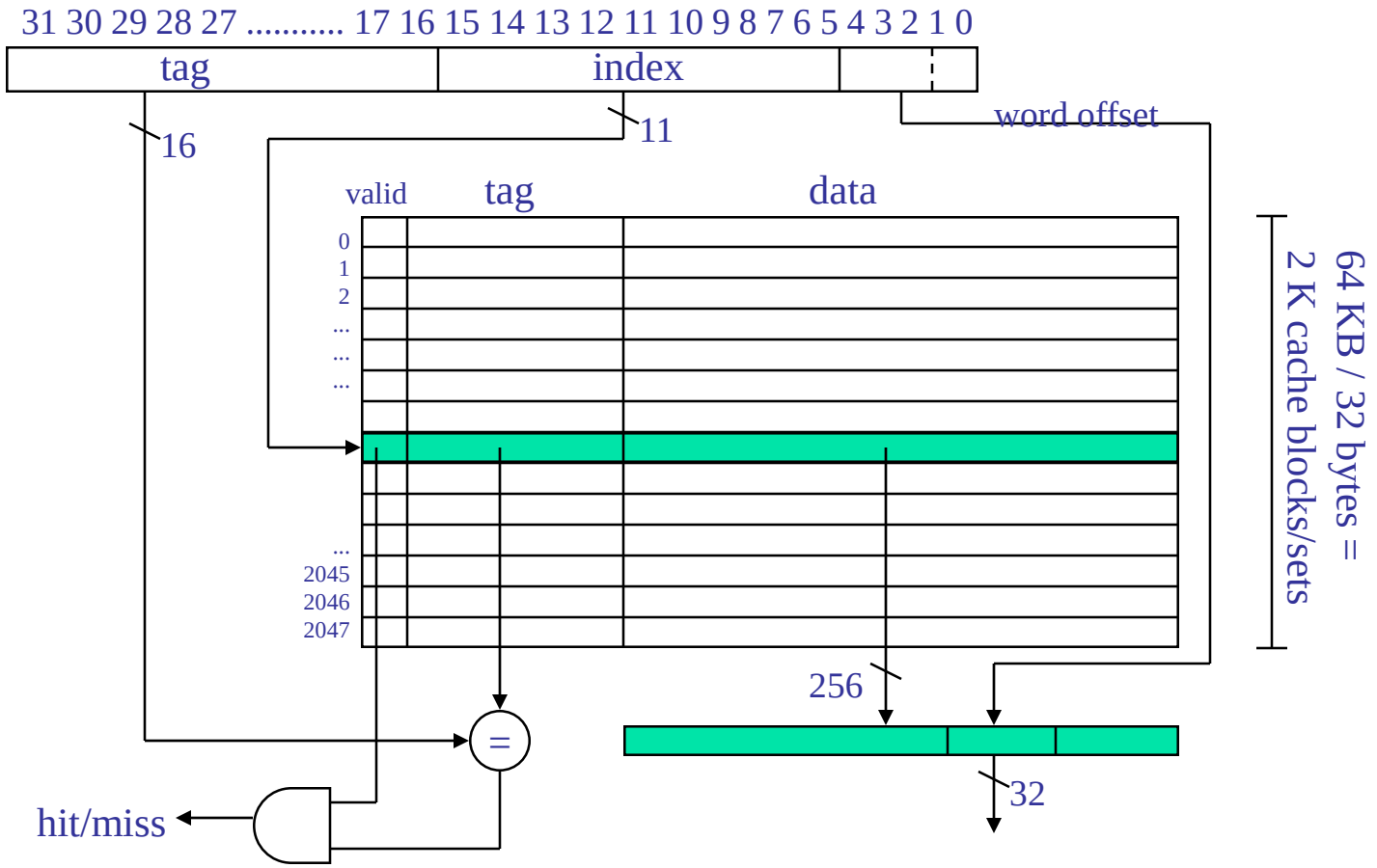
If line length is n bytes, the low-order $\log_2 n$ bits of a Byte-address give the offset of address within a line.

The next group of bits is the index – this indexes into the number of cache sets a cache can hold. (We also use set to refer to the same thing.)

The remaining bits are the tag.

Putting it all together

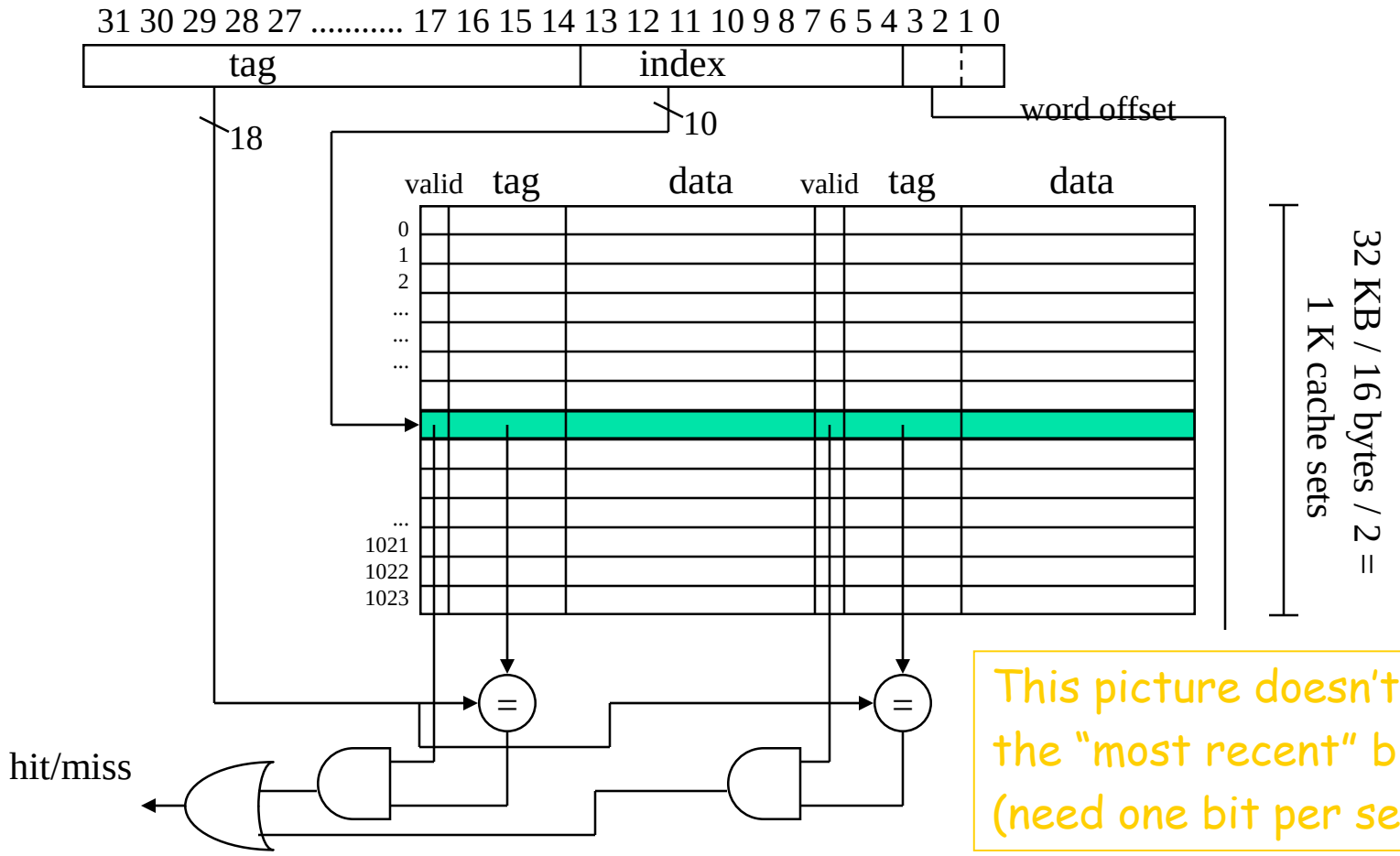
64 KB cache, direct-mapped, 32-byte cache block



ILLINOIS

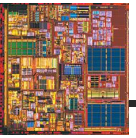
A set associative cache

32 KB cache, 2-way set-associative, 16-byte blocks



This picture doesn't show the "most recent" bit (need one bit per set)

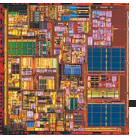
ILLINOIS



Sample Problem

A cache has a capacity of 32 KB and 256-byte lines. On a machine with a 32-bit virtual address space, how many bits long are the tag, set, and offset fields for

- A direct-mapped implementation?
- A four-way set-associative implementation?
- A fully-associative implementation?

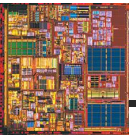


Sample Problem Version II

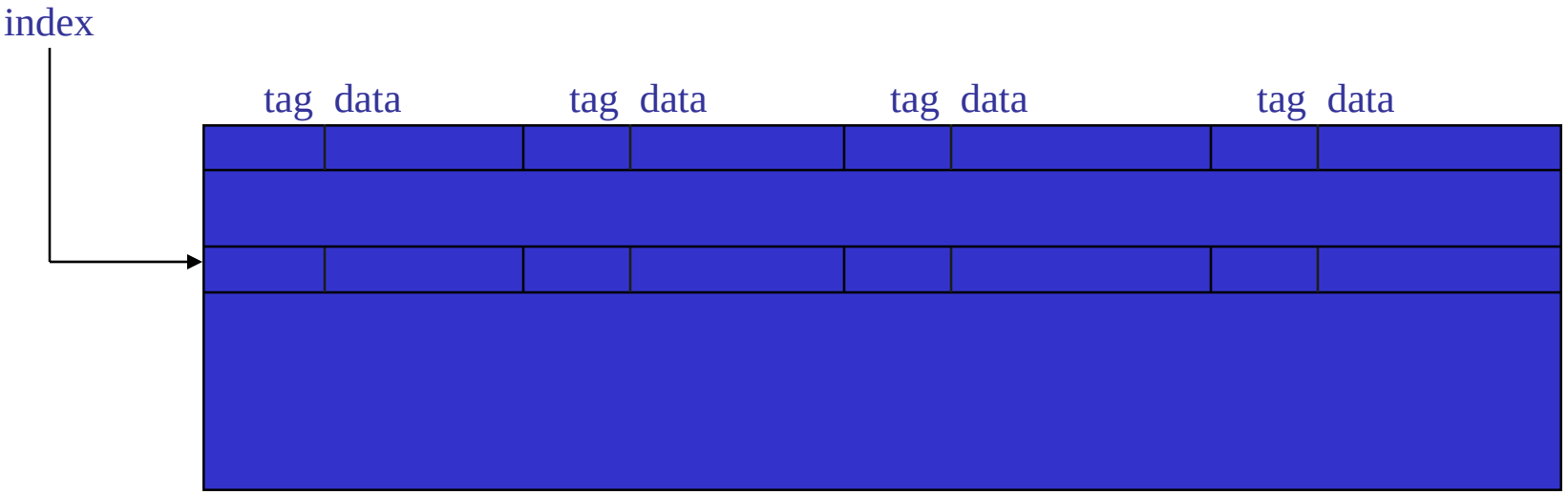
A cache has a capacity of 32 KB and 256-byte in a set. On a machine with a 32-bit address space, how many bits long are the tag, set, and offset fields for

- A direct-mapped implementation?
- A four-way set-associative implementation?
- A eight-way set-associative implementation?

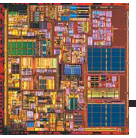
Another example



- 16 KB, 4-way set-associative cache, 32-bit address, byte-addressable memory, 32-byte cache blocks/lines
- how many tag bits?
- Where would you find the word at address 0x200356A4?



ILLINOIS



READ P&H 5.1, 5.2, 5.3
FINISH MP1, START MP2!