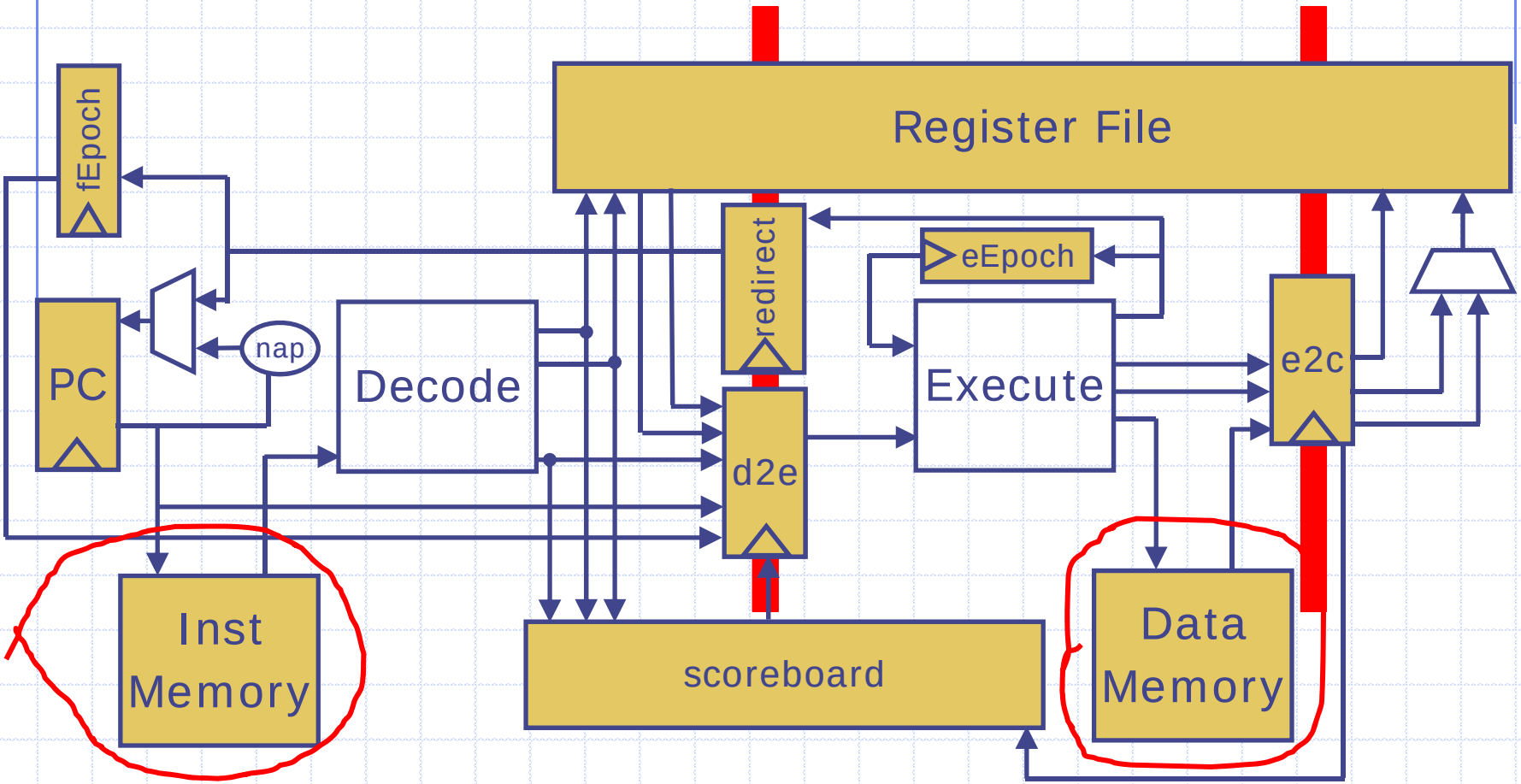
Constructive Computer Architecture

Realistic Memories and Caches

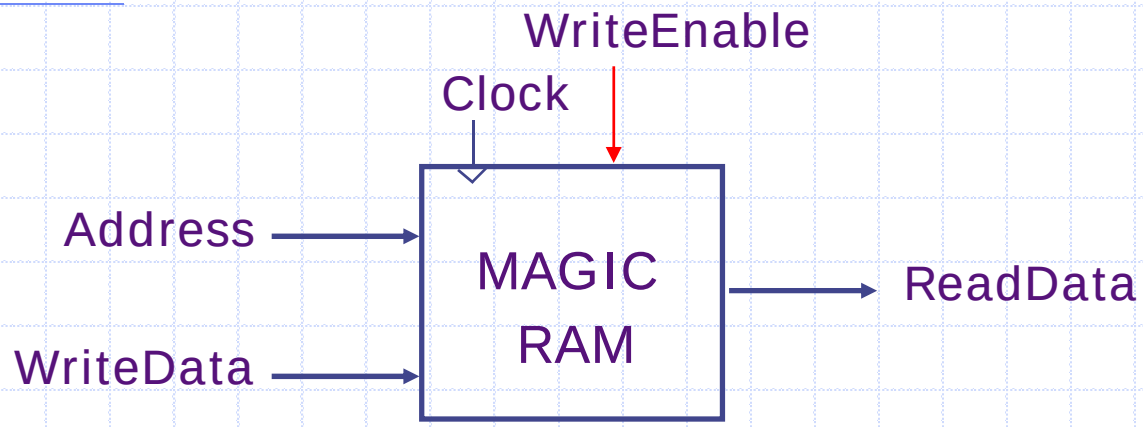
Arvind
Computer Science & Artificial Intelligence Lab.
Massachusetts Institute of Technology

Multistage Pipeline



The use of magic memories (combinational reads) makes such design unrealistic

Magic Memory Model

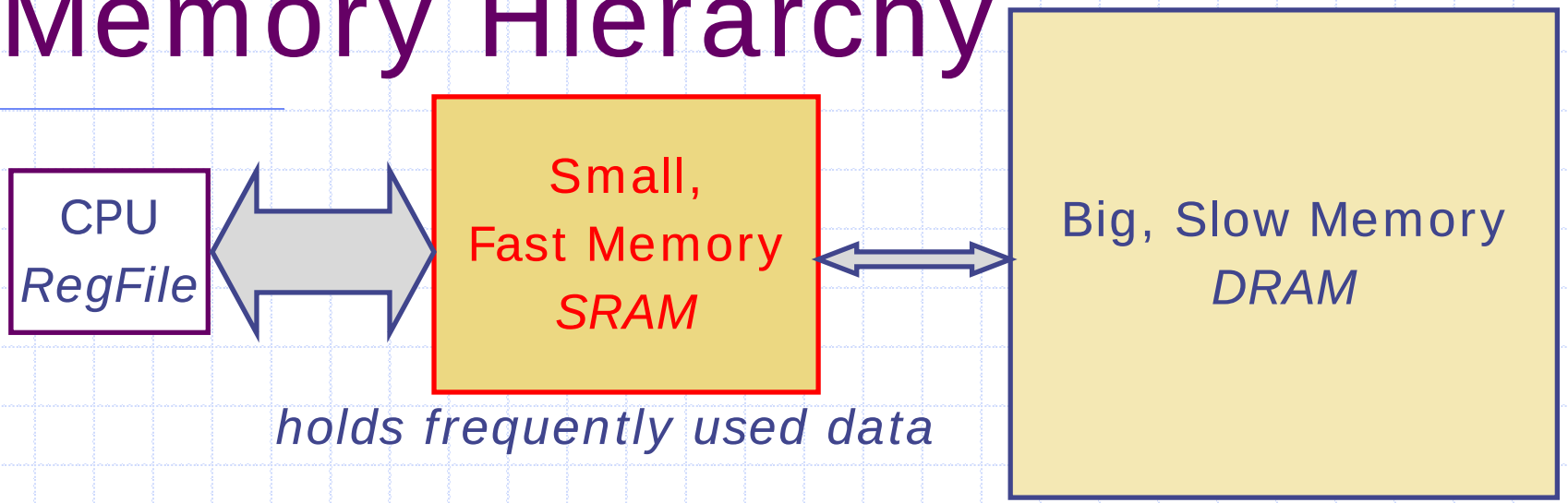


- Reads and writes are always completed in one cycle
  - a Read can be done any time (i.e. combinational)
  - If enabled, a Write is performed at the rising clock edge

    (*the write address and data must be stable at the clock edge)*

In a real DRAM the data will be available several cycles after the address is supplied

Memory Hierarchy



size: RegFile << SRAM << DRAM

latency: RegFile << SRAM << DRAM

bandwidth: on-chip >> off-chip

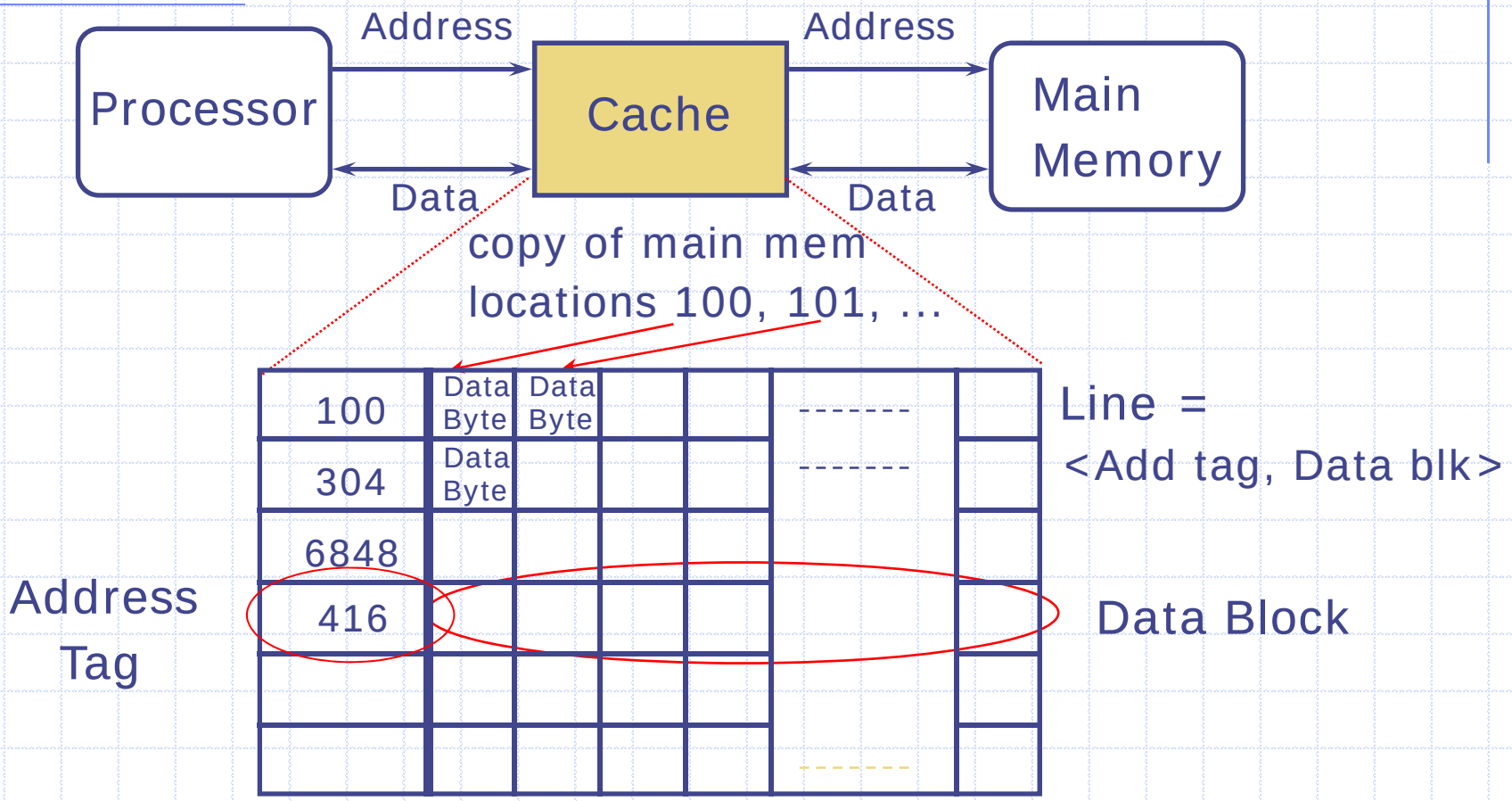
why?

On a data access:

hit (data $\in$ fast memory) $\Rightarrow$ low latency access

miss (data $\notin$ fast memory) $\Rightarrow$ long latency access *(DRAM)*

Inside a Cache



How many bits are needed for the tag?

Enough to uniquely identify the block

Cache Read

Search cache tags to find match for the processor generated address

Found in cache
a.k.a. hit

Not in cache
a.k.a. miss

Return copy of data from cache

Read block of data from Main Memory – may require writing back a cache line

Wait ...

Which line do we replace?

Return data to processor and update cache

Write behavior

- On a write hit
  - Write-through: write to both cache and the next level memory
  - write-back: write only to cache and update the next level memory when line is evacuated
- On a write miss
  - Allocate – because of multi-word lines we first fetch the line, and then update a word in it
  - No allocate – word modified in memory

Cache Line Size

- A cache line usually holds more than one word
  - Reduces the number of tags and the tag size needed to identify memory locations
  - Spatial locality: Experience shows that if address x is referenced then addresses x+1, x+2 etc. are very likely to be referenced in a short time window
    - consider instruction streams, array and record accesses
  - Communication systems (e.g., bus) are often more efficient in transporting larger data sets

Types of misses

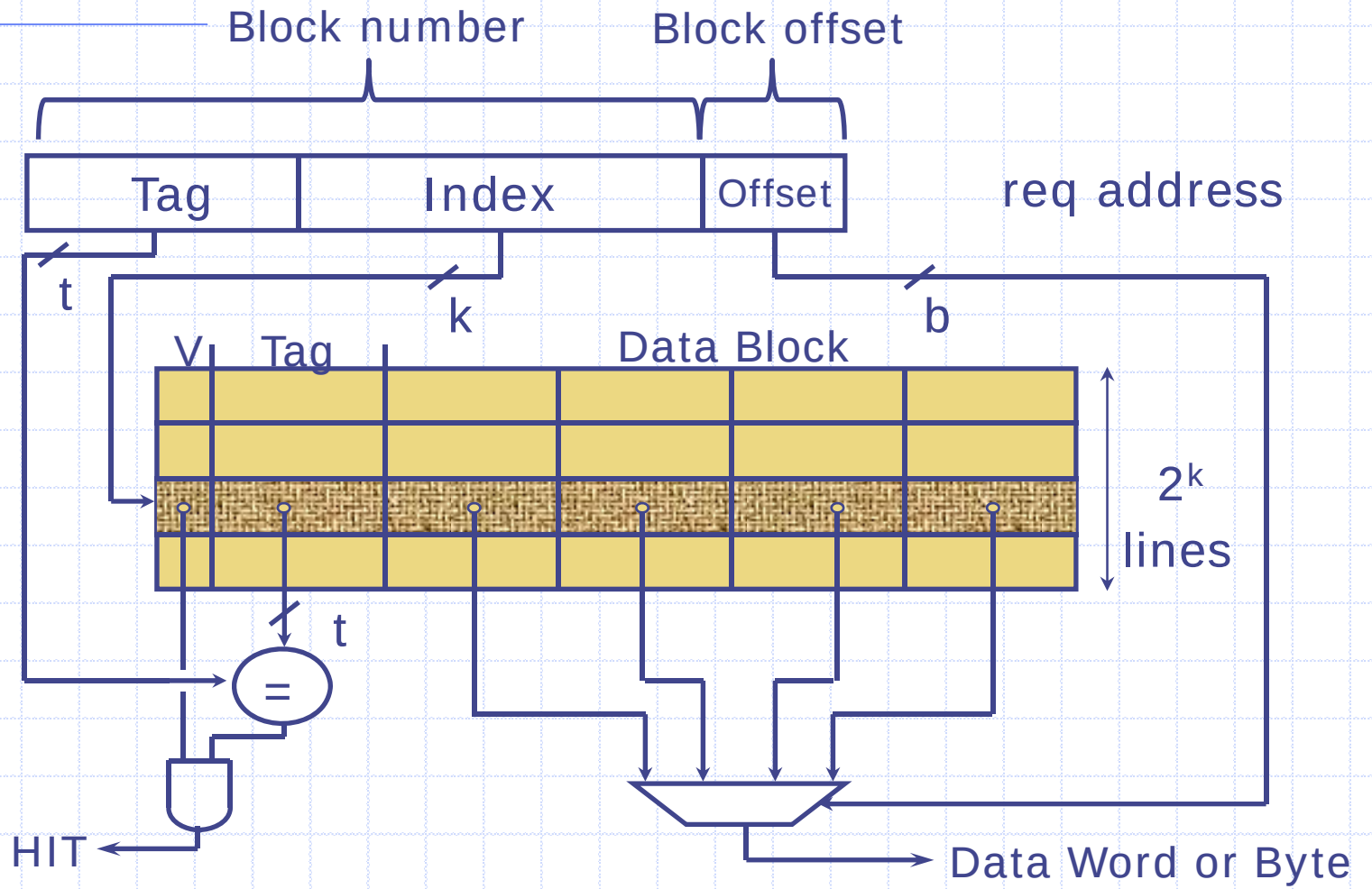
- Compulsory misses (cold start)
  - First time data is referenced
  - Become insignificant if billions of instructions are run
- Capacity misses
  - *Working set* is larger than cache size
  - Solution: increase cache size
- Conflict misses
  - Usually multiple memory locations are mapped to the same cache location to simplify implementations
  - Thus it is possible that the designated cache location is full while there are empty locations in the cache.
  - Solution: Set-Associative Caches

Internal Cache Organization

- Cache designs restrict where in cache a particular address can reside
  - *Direct mapped:* An address can reside in exactly one location in the cache. The cache location is typically determined by the lowest order address bits
  - *n-way Set associative:* An address can reside in any of the a set of n locations in the cache. The set is typically determine by the lowest order address bits

Direct-Mapped Cache

The simplest implementation

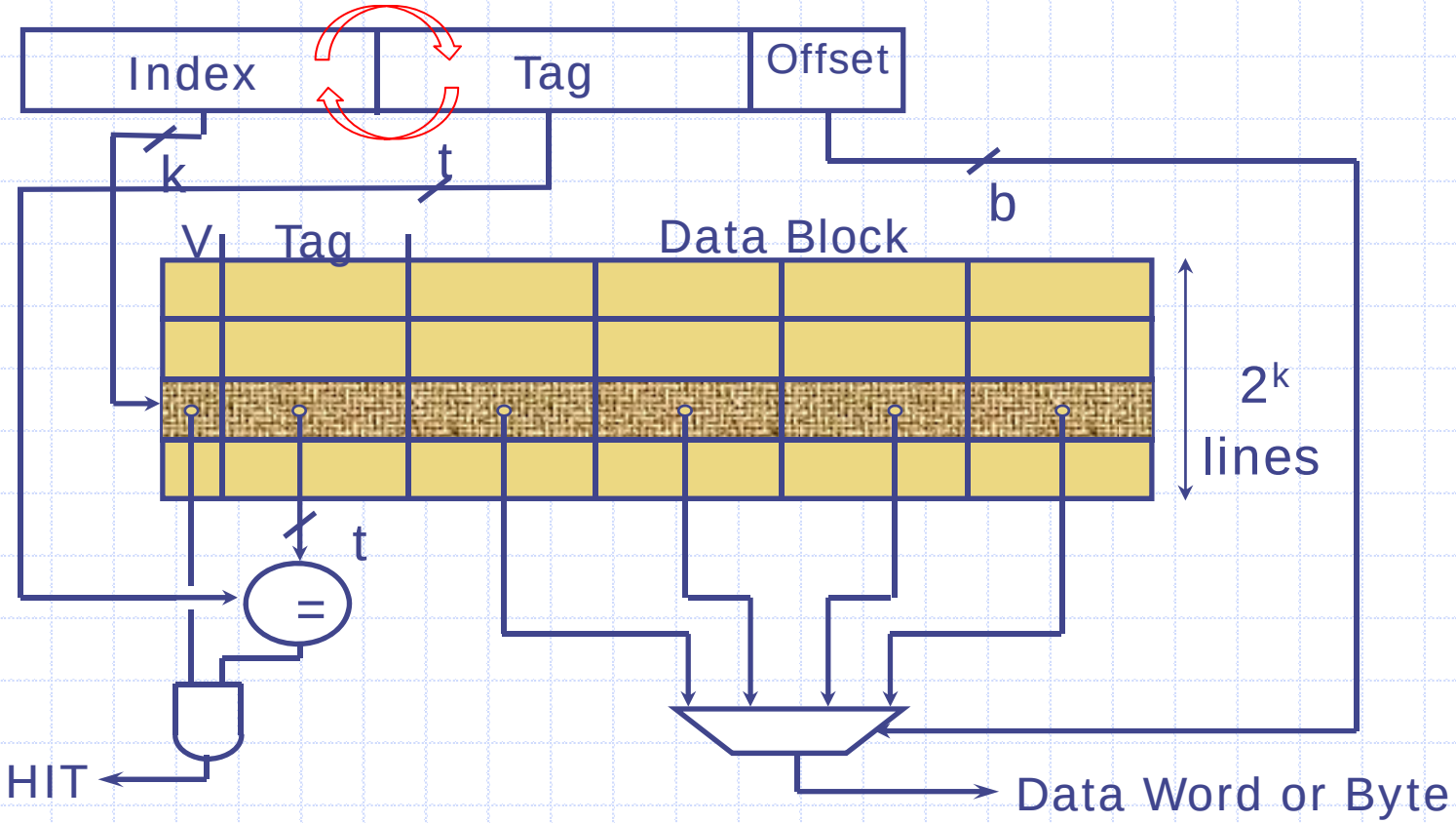


What is a bad reference pattern?

Strided = size of cache

Direct Map Address Selection

higher-order vs. lower-order address bits



Why higher-order bits as tag may be undesirable?

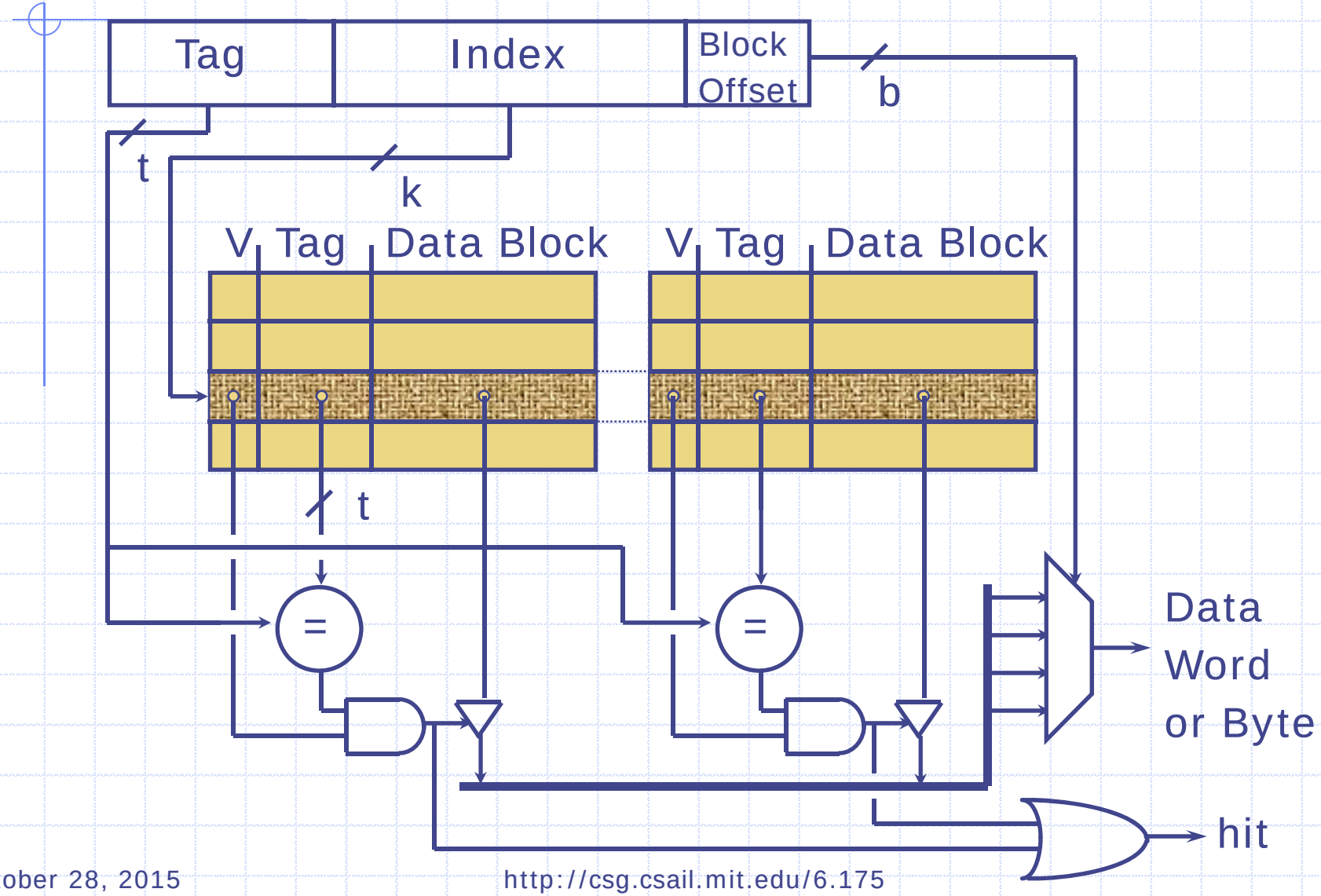
Spatially local blocks conflict

Reduce Conflict Misses

Memory time =
Hit time + Prob(miss) * Miss penalty

- Associativity: Reduce conflict misses by allowing blocks to go to several sets in cache
  - 2-way set associative: each block can be mapped to one of 2 cache sets
  - Fully associative: each block can be mapped to any cache frame

2-Way Set-Associative Cache



Replacement Policy

- In order to bring in a new cache line, usually another cache line has to be thrown out. Which one?
  - No choice in replacement if the cache is direct mapped
- Replacement policy for set-associative caches
  - One that is not dirty, i.e., has not been modified
    - In I-cache all lines are clean
    - In D-cache if a dirty line has to be thrown out then it must be written back first
  - Least recently used?
  - Most recently used?
  - Random?

How much is performance affected by the choice?

Difficult to know without benchmarks and quantitative measurements

Blocking vs. Non-Blocking cache

- Blocking cache:
  - At most one outstanding miss
  - Cache must wait for memory to respond
  - Cache does not accept requests in the meantime
- Non-blocking cache:
  - Multiple outstanding misses
  - Cache can continue to process requests while waiting for memory to respond to misses

We will first design a write-back, Write-miss allocate, Direct-mapped, blocking cache