S. Di Girolamo [digirols@inf.ethz.ch]

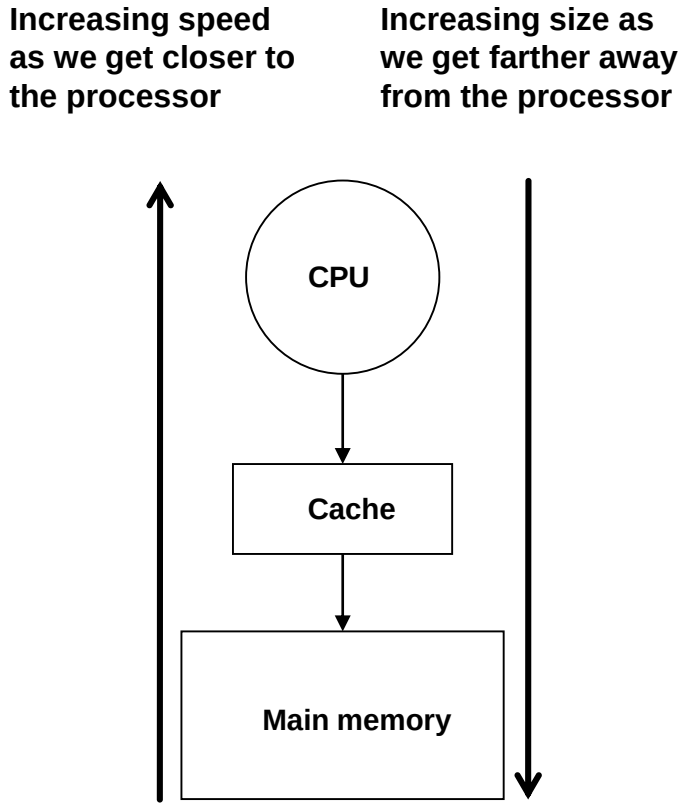
Caches

Design of Parallel and High-Performance Computing – Recitation Session

Slides credits: Pavan Balaji, Torsten Hoefler
https://htor.inf.ethz.ch/teaching/mpi_tutorials/ppopp13/2013-02-24-ppopp-mpi-basic.pdf

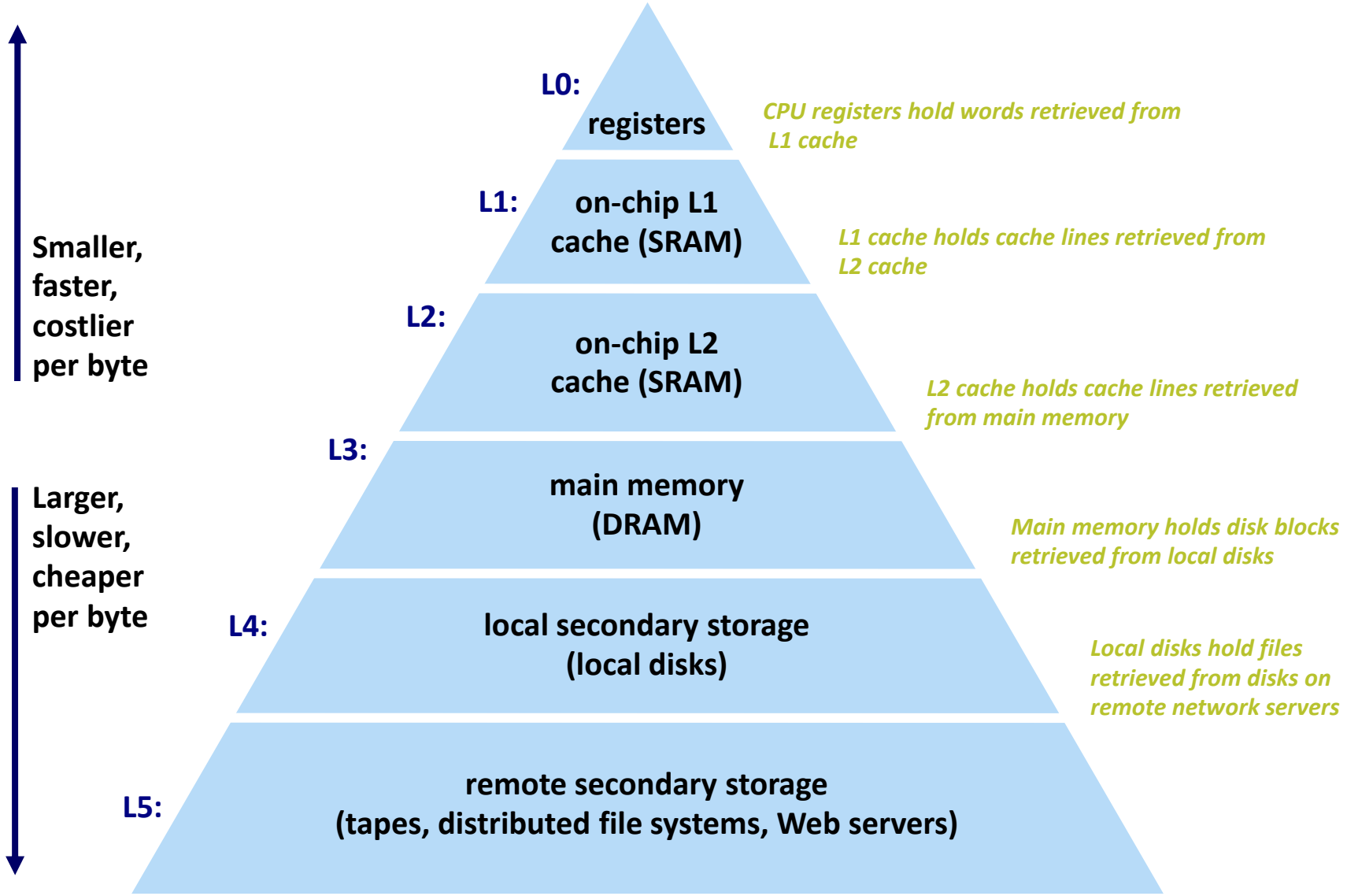
Idea

- Feasible to have small amount of fast memory and/or large amount of slow memory.
- Want
  - Size advantage of DRAM
  - Speed advantage of SRAM.
- CPU looks in cache for data it seeks from main memory.
- If data not there it retrieves it from main memory.
- If the cache is able to service "most" CPU requests then effectively we will get speed advantage of cache.
- All addresses in cache are also in memory

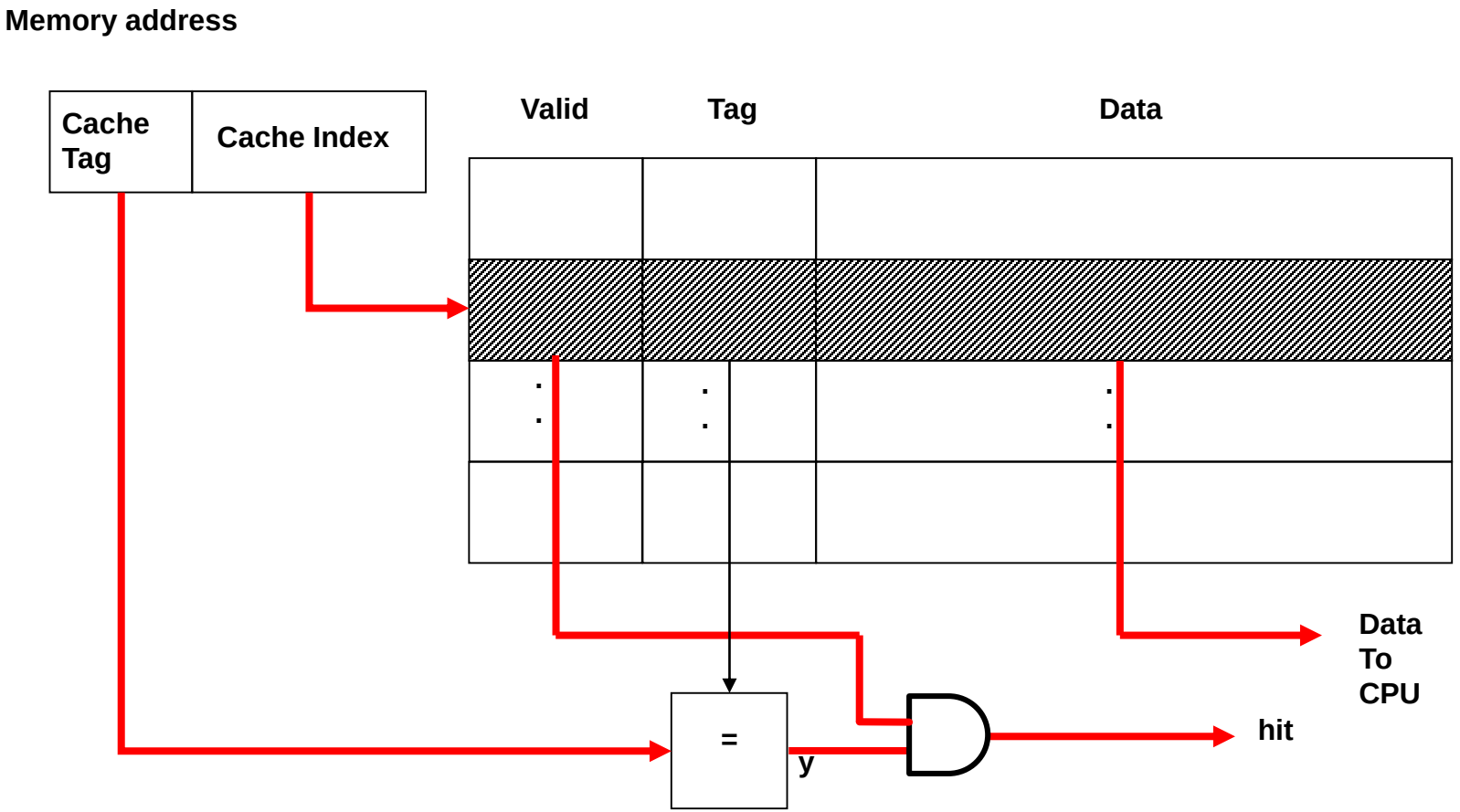


https://www.cc.gatech.edu/~rama/CS2200-External/ppt-slides/Chapter09.pptx

Typical Memory Hierarchy



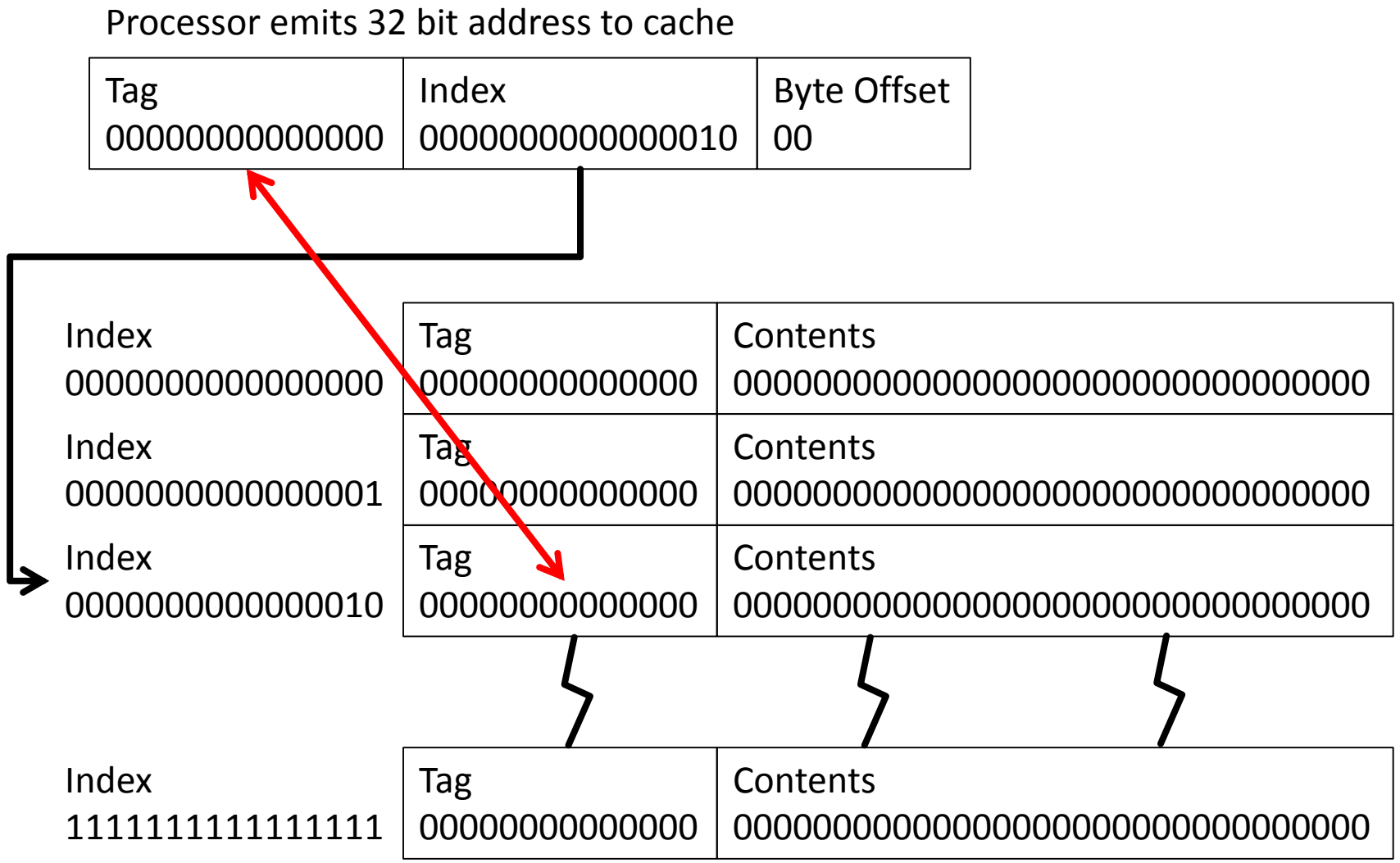
Direct Mapping



https://www.cc.gatech.edu/~rama/CS2200-External/ppt-slides/Chapter09.pptx

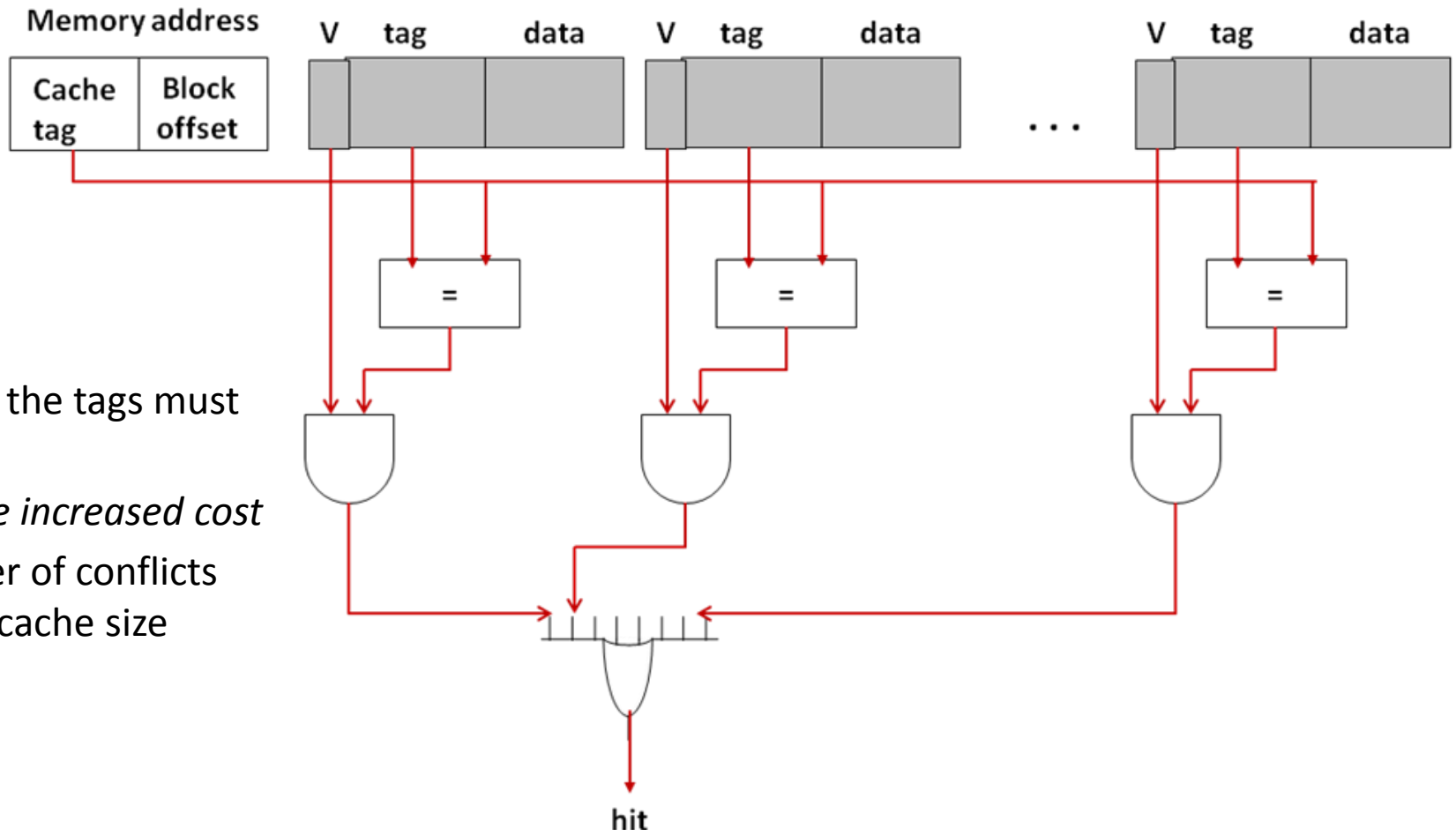
Why do we need the validity bit?

Assume computer is turned on and every location in cache is zero. What can go wrong?



https://www.cc.gatech.edu/~rama/CS2200-External/ppt-slides/Chapter09.pptx

Fully Associative

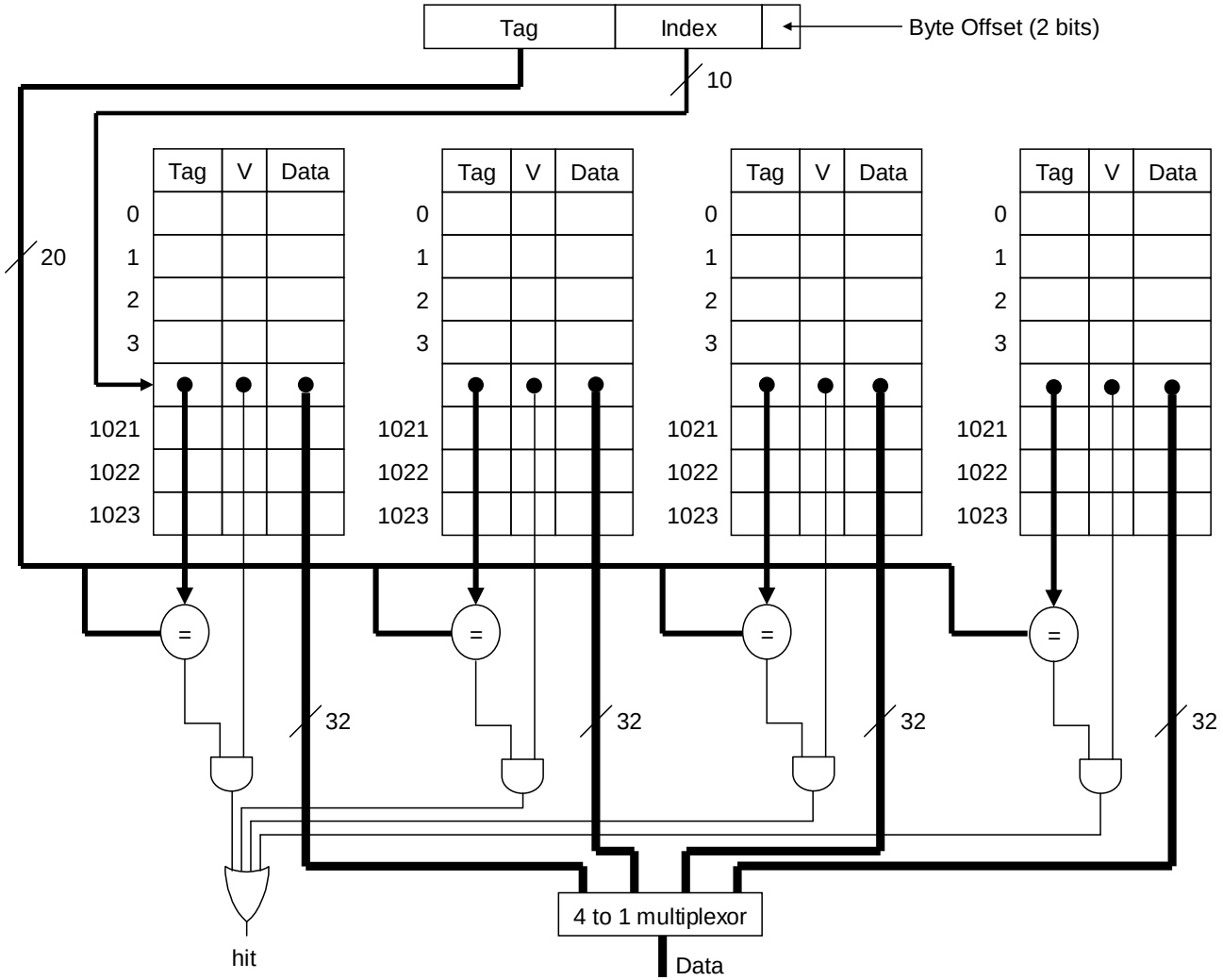


- **Fully Associative**
  - A block can go anywhere
  - To check if a block is cached, all the tags must be compared!

    Increased HW complexity, hence increased cost
  - Diminishing returns: the number of conflicts decreases when increasing the cache size

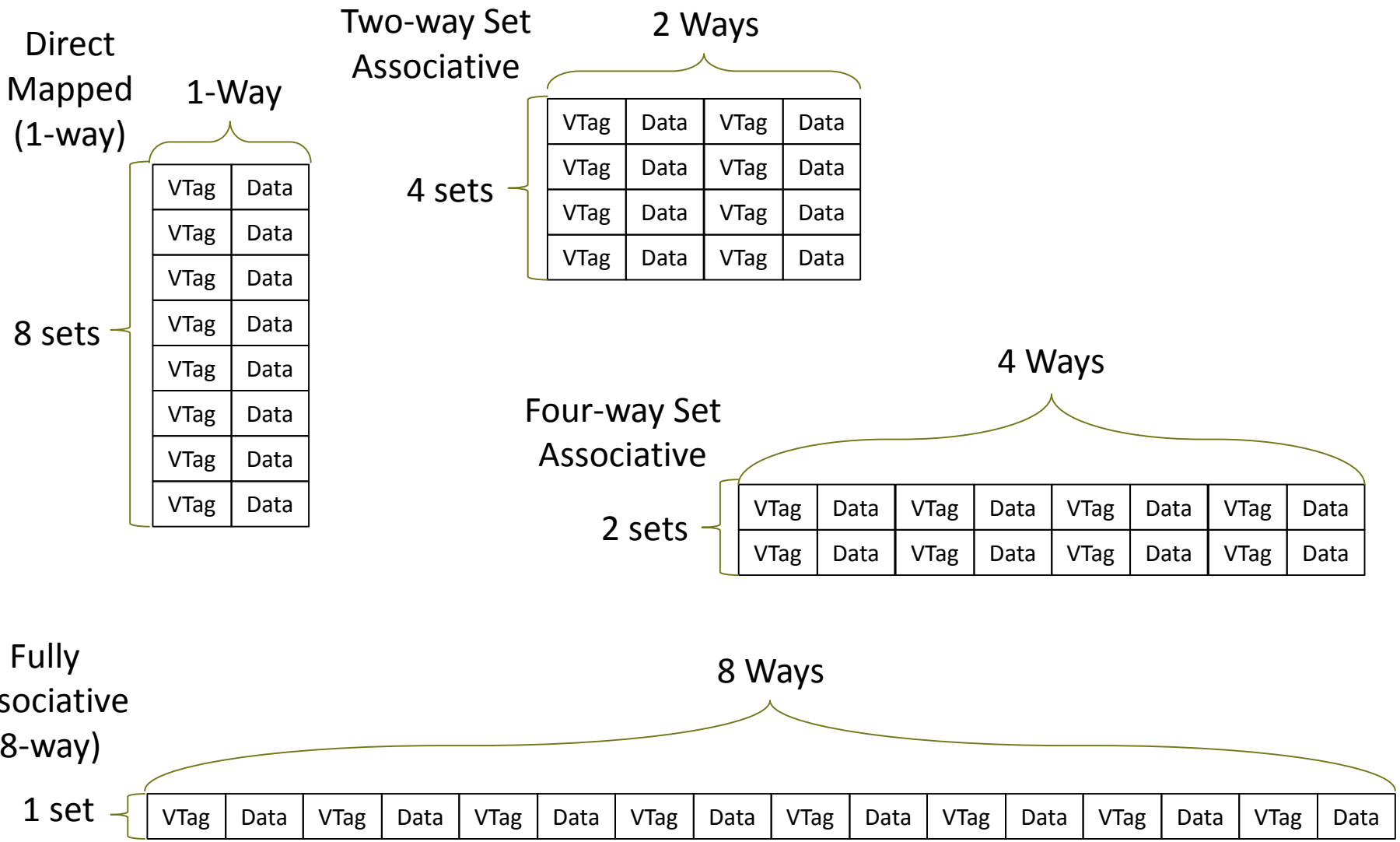
https://www.cc.gatech.edu/~rama/CS2200-External/ppt-slides/Chapter09.pptx

Set Associative



https://www.cc.gatech.edu/~rama/CS2200-External/ppt-slides/Chapter09.pptx

Extremes of set associativity



https://www.cc.gatech.edu/~rama/CS2200-External/ppt-slides/Chapter09.pptx

Quiz

- How caches enable the exploiting of spatial locality?

- How to make sure that a newly allocated memory region starts at cacheline?

How to allocate block-aligned memory?

Synopsis

#include <stdlib.h>
int posix_memalign(void **memptr, size_t alignment, size_t size);

Description

The function **posix_memalign**() allocates *size* bytes and places the address of the allocated memory in **memptr*. The address of the allocated memory will be a multiple of *alignment*, which must be a power of two and a multiple of *sizeof(void *)*. If *size* is 0, then **posix_memalign**() returns either NULL, or a unique pointer value that can later be successfully passed to **free**(3).

Return value

posix_memalign() returns zero on success, or one of the error values listed in the next section on failure. Note that *errno* is not set.

The 3C's

- **Compulsory misses**
  - cold start
  - don't have a choice
  - **Can we reduce the number of compulsory misses?**

    Increasing block size can reduce the number of distinct blocks that are requested

- **Capacity misses**
  - cache is much smaller than total addressable memory

- **Conflict misses**
  - collision within a set
  - requested block was thrown out when some other block wanted to occupy the same position
  - can reduce by increasing associativity (each block has more options)

Exercise 1

- **Blackboard**

Replacement

- **How do we choose *victim*?**
  - Verbs: *Victimize, evict, replace, cast out*
- **Many policies are possible**
  - FIFO (first-in-first-out)
  - LRU (least recently used), pseudo-LRU
  - LFU (least frequently used)
  - NMRU (not most recently used)
  - NRU
  - Pseudo-random (yes, really!)
  - Optimal
  - Etc

Optimal: [Belady, IBM Systems Journal, 1966]

- **Evict block with longest reuse distance**
  - i.e. next reference to block is farthest in future
  - Requires knowledge of the future!

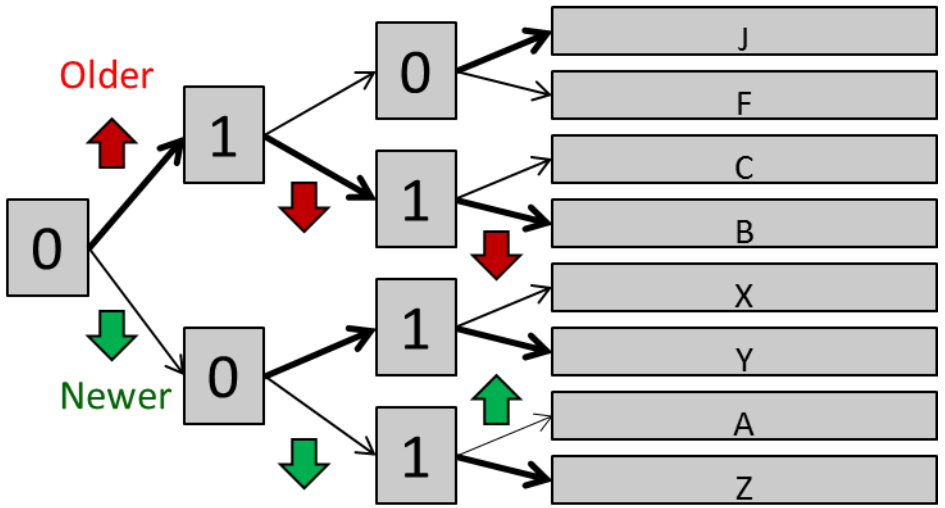
https://ece752.ece.wisc.edu/lect11-cache-replacement.pdf

Least-Recently Used

- **For a=2, LRU is equivalent to NMRU**
  - Single bit per set indicates LRU/MRU
  - Set/clear on each access
- **For a>2, LRU is difficult/expensive**
  - Timestamps? How many bits?

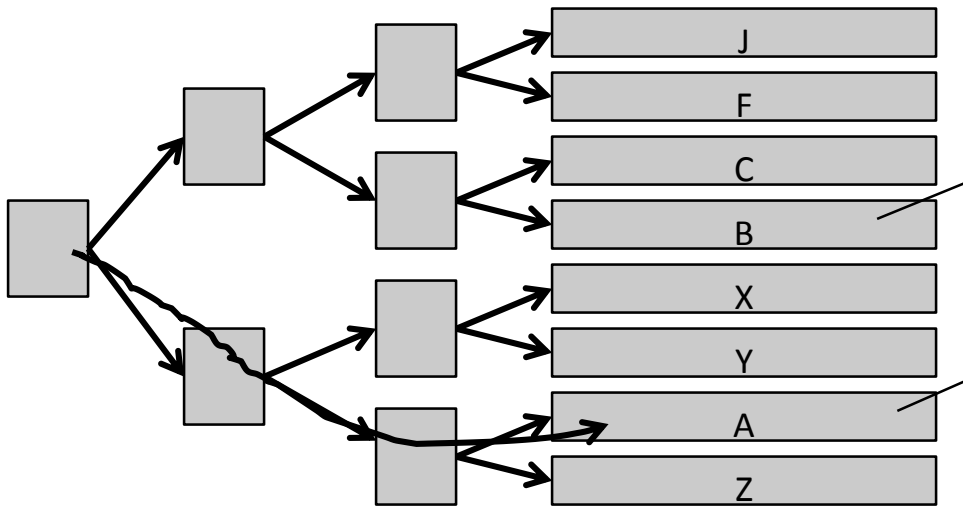
    Must find min timestamp on each eviction
  - Sorted list? Re-sort on every access?

Practical Pseudo-LRU



https://ece752.ece.wisc.edu/lect11-cache-replacement.pdf

Practical Pseudo-LRU In Action



J Y X Z B C F A

011: PLRU
Block B is here

110: MRU
block is here

https://ece752.ece.wisc.edu/lect11-cache-replacement.pdf

Exercise 2

- **Blackboard**