# EEL 5764 Computer Architecture

**Sandip Ray**
Department of Electrical and Computer Engineering
University of Florida

## **Lecture 10:** Cache Optimizations

# Announcements

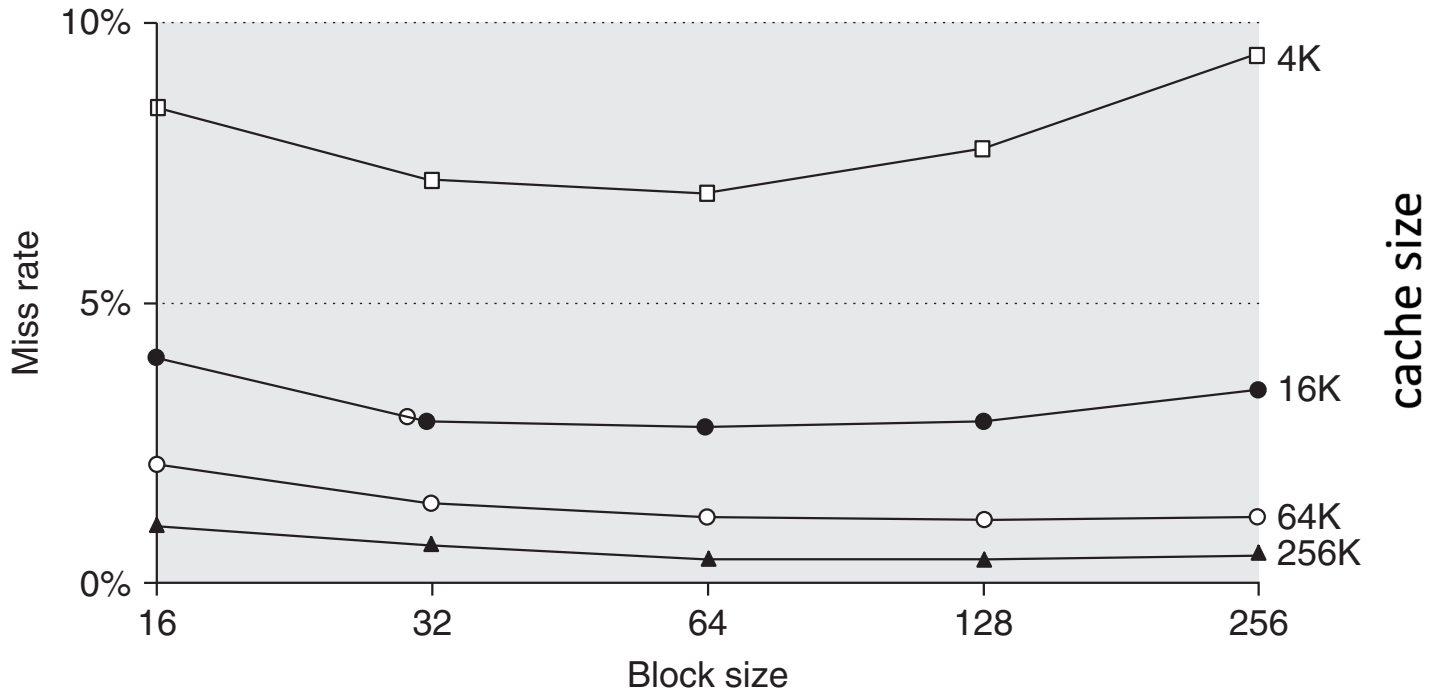
- **Corrected a typo in the homework assignment schedule.** Look at the updated schedule in the Web page
- **Added a tentative schedule of lectures.** Note that this is subject to change based on the pace of the class.
- **No extension on Problem Set 1.** But it's not graded anyhow and we'll only look at the problem sets (rather than HWs) if you're borderline between two grades.
- Please post your questions on canvas.
- **Implementation projects can be in groups of 2 (minimum) or 4 maximum.** Surveys by no more than 2.
  - → Please start the process of forming your teams. Feel free to use Canvas discussions for that.
- Surveys will be graded much more stringently than implementation projects

# Six Basic Cache Optimizations

- Larger block size
  - → Reduces compulsory misses
  - → Increases capacity and conflict misses, increases miss penalty
- Larger total cache capacity to reduce miss rate
  - → Increases hit time, increases power consumption
- Higher associativity
  - → Reduces conflict misses
  - → Increases hit time, increases power consumption
- Multi-level cache to reduce miss penalty
  - → Reduces overall memory access time
- Giving priority to read misses over writes
  - → Reduces miss penalty
- Avoiding address translation in cache indexing
  - → Reduces hit time

# Optimization 1 – Larger Block Size

- Reduce compulsory misses due to spatial locality
- May increase conflict/capacity misses
- Increase miss penalty



# Optimization 1 – Larger Block Size

- Reduce compulsory misses due to spatial locality
- May increase conflict/capacity misses
- Increase miss penalty

| | | Cache size | | | |
|---|---|---|---|---|---|
| Block size | Miss penalty | 4K | 16K | 64K | 256K |
| 16 | 82 | 8.027 | 4.231 | 2.673 | 1.894 |
| 32 | 84 | **7.082** | 3.411 | 2.134 | 1.588 |
| 64 | 88 | 7.160 | **3.323** | **1.933** | **1.449** |
| 128 | 96 | 8.469 | 3.659 | 1.979 | 1.470 |
| 256 | 112 | 11.651 | 4.685 | 2.288 | 1.549 |

AMAT

# Optimization 3 – Higher Associativity

- Reduces conflict misses
- Increases hit time
- Increases power consumption

| | Associativity | | | |
|---|---|---|---|---|
| Cache size (KB) | 1-way | 2-way | 4-way | 8-way |
| 4 | 3.44 | 3.25 | 3.22 | **3.28** |
| 8 | 2.69 | 2.58 | 2.55 | **2.62** |
| 16 | 2.23 | **2.40** | **2.46** | **2.53** |
| 32 | 2.06 | **2.30** | **2.37** | **2.45** |
| 64 | 1.92 | **2.14** | **2.18** | **2.25** |
| 128 | 1.52 | **1.84** | **1.92** | **2.00** |
| 256 | 1.32 | **1.66** | **1.74** | **1.82** |
| 512 | 1.20 | **1.55** | **1.59** | **1.66** |

# Optimization 4 – Multilevel Cache

- L1 hit time affects CPU speed
  → Small and fast L1
- Speed of L2 affects L1 miss penalty
  → Large L2 with higher associativity

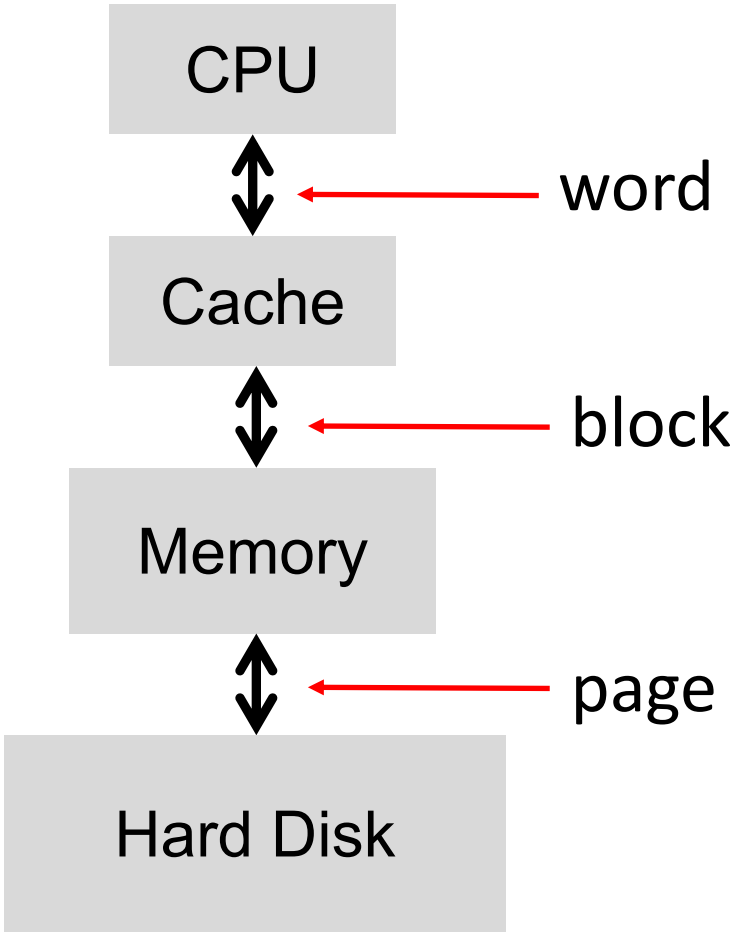
# Optimization 5 – Giving Priority to Read Misses over Writes

- Reduces miss penalty. See example below.

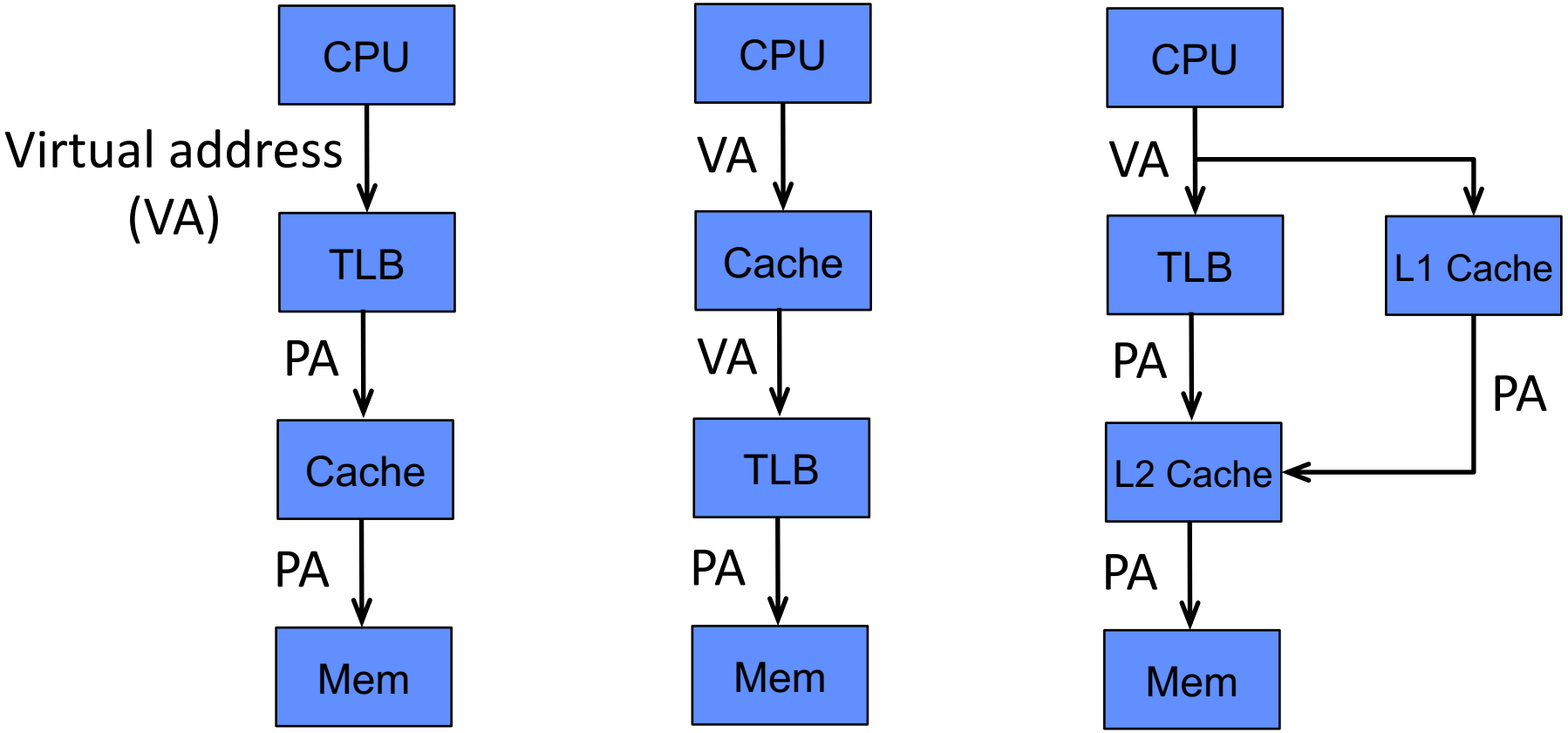
```
SW R3, 512(R0)     ;M[512] ← R3     (cache index 0)
LW R1, 1024(R0)    ;R1 ← M[1024]    (cache index 0)
LW R2, 512(R0)     ;R2 ← M[512]     (cache index 0)
```

- Write-through cache with write buffers suffers from RAW conflicts with main memory reads on cache misses:
  - → Write buffer holds updated data needed for the read.
  - → Alt #1 – wait for the write buffer to empty, increasing read miss penalty (in old MIPS 1000 by 50% ).
  - → Alt #2 – Check write buffer contents before a read; if no conflicts, let the memory read go first.

# Optimization 6 – Avoid Address Translation during Cache Indexing to Reduce Hit Time



# Optimization 6 – Avoid Address Translation during Cache Indexing to Reduce Hit Time




Conventional organization

Virtually addressed cache

Virtually addressed physically tagged

# Advanced Cache Optimizations

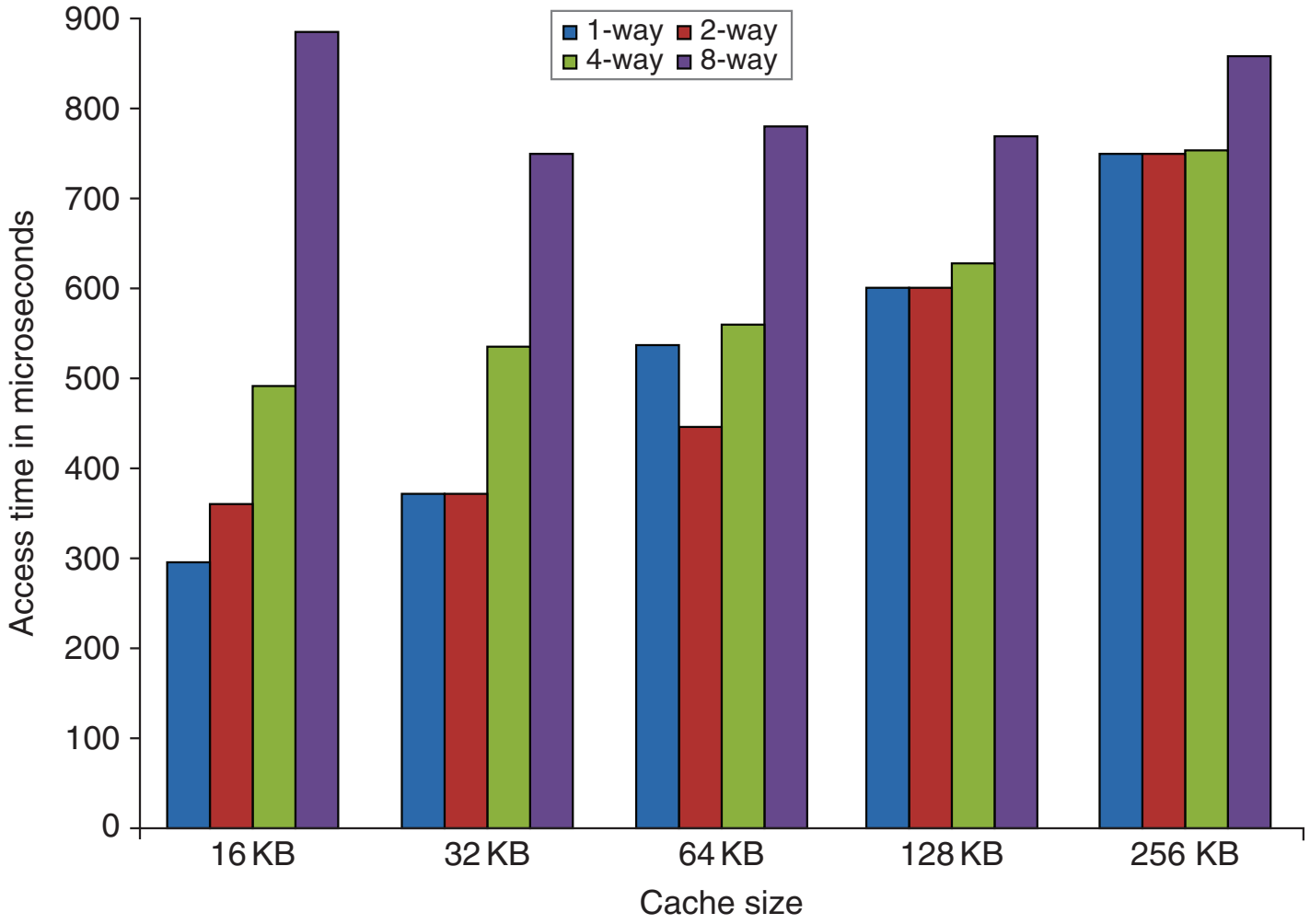
# Basic Ideas

- Reduce hit time, miss rate, and miss penalty
- Increase cache bandwidth
- Reduce miss rate/penalty via parallelism

# Opt 1 – Small and Simple L1 Caches

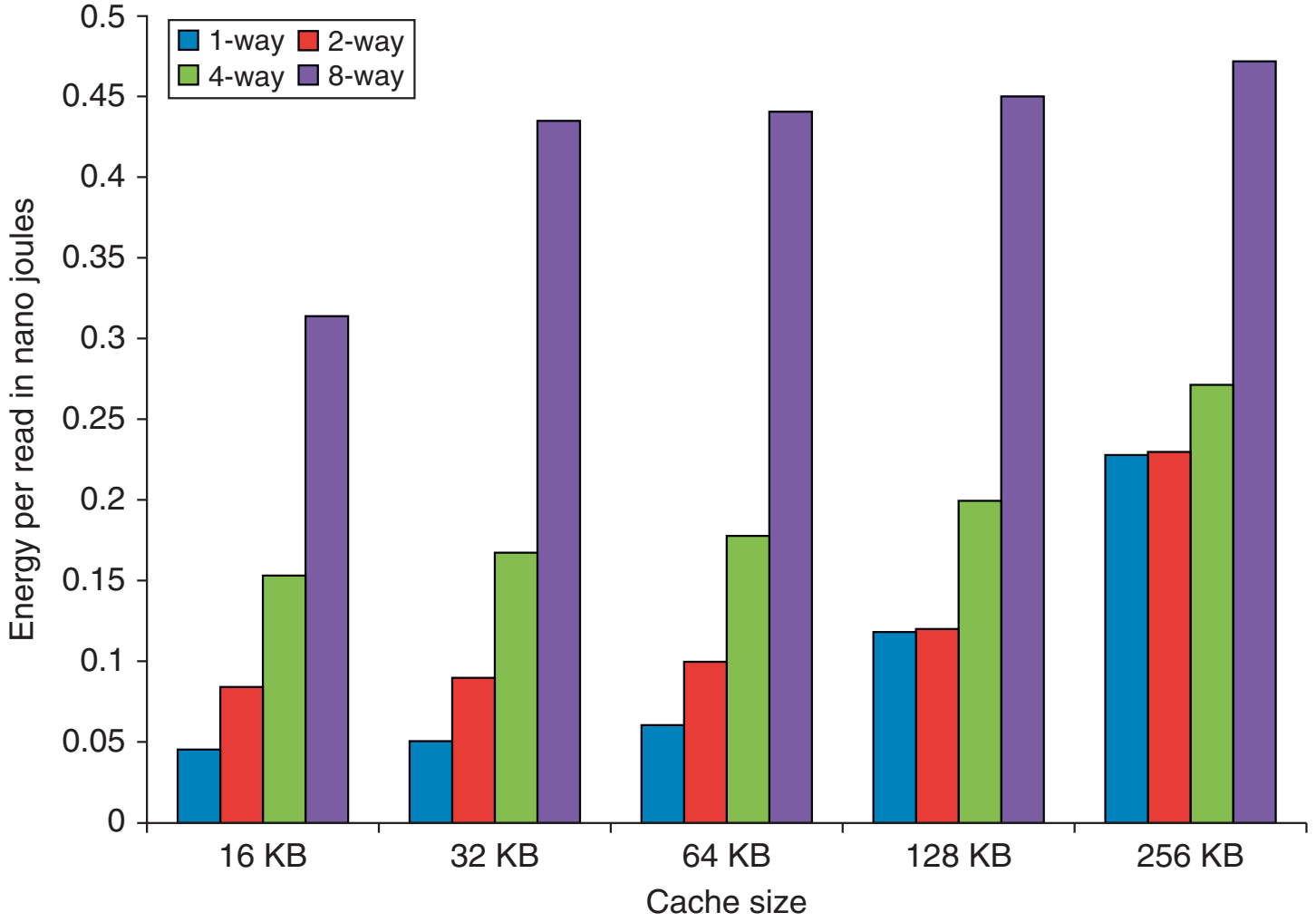
- Critical timing path in cache hit:
  - → addressing tag memory, then
  - → comparing tags, then
  - → selecting correct set
- Direct-mapped caches can overlap tag comparison and transmission of data
- Lower associativity reduces power because fewer cache lines are accessed. However,
  - → Associativity used to increase size of virtually indexed cache
  - → Higher associativity reduces conflict misses due to multithreading

# Opt 1 – L1 Size and Associativity



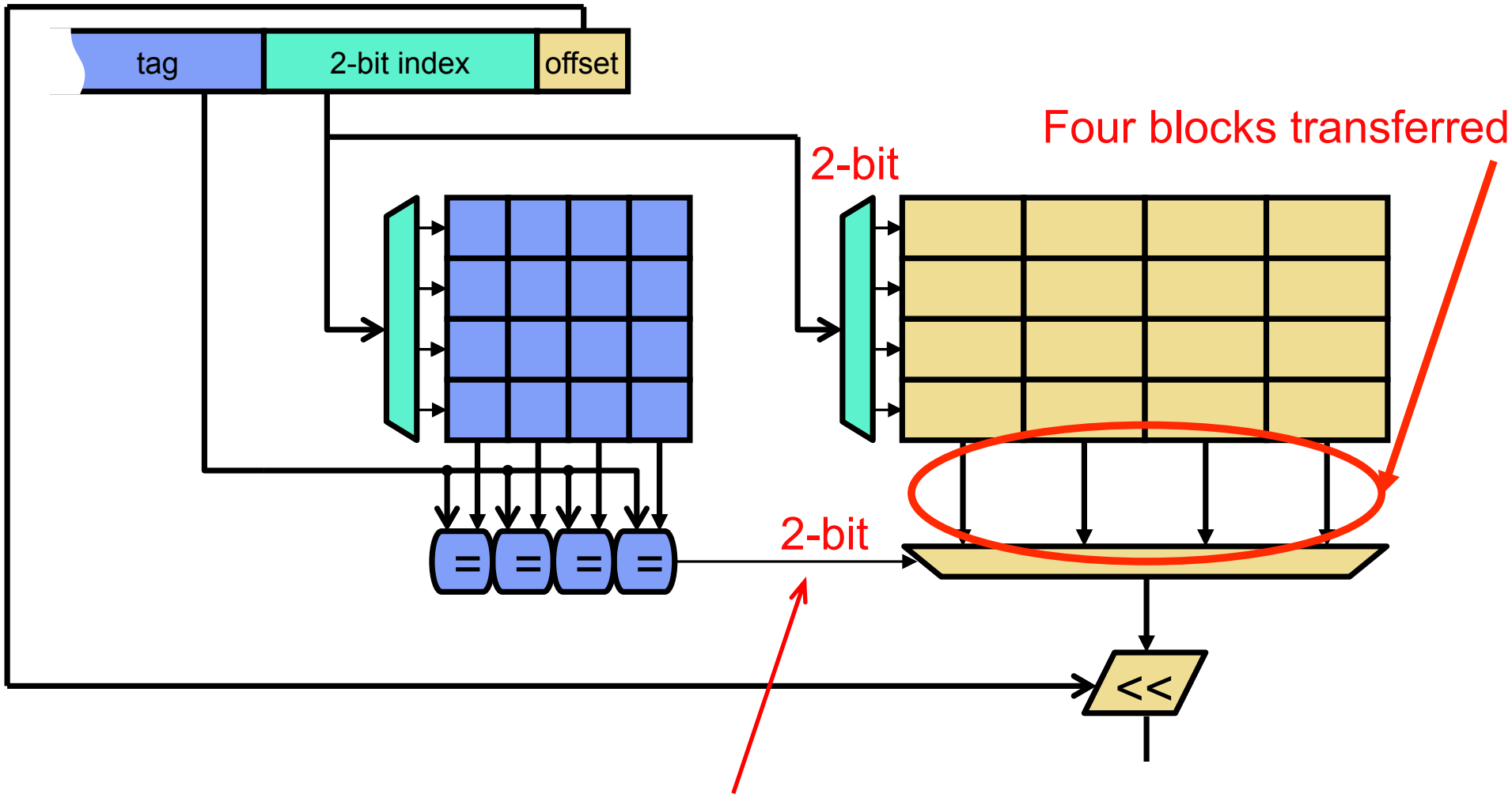

Access time increases as size & associativity increases

# Opt 1 – L1 Size and Associativity



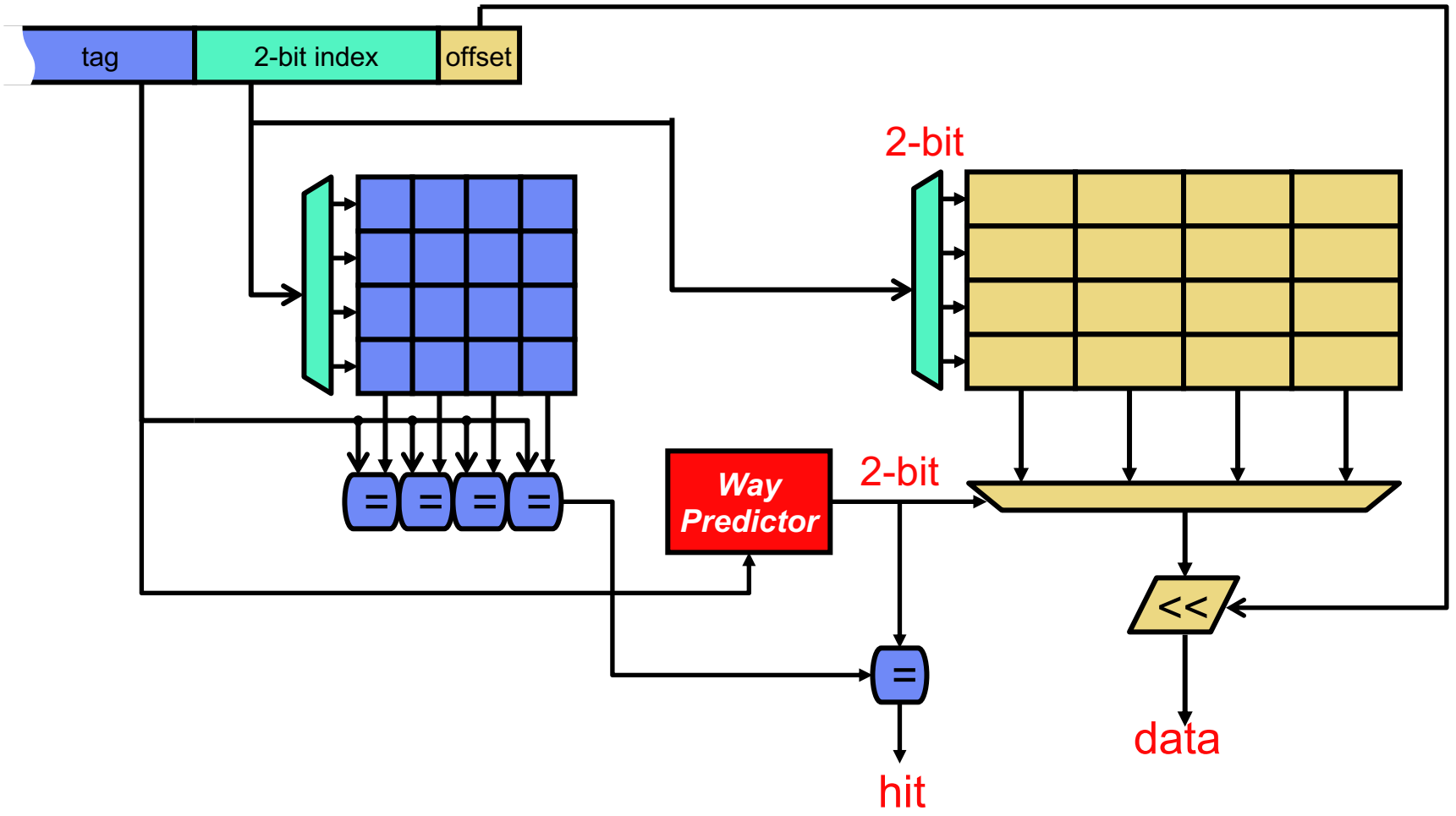

Energy per read increases as size & associativity increases

# Opt 2 – Way Prediction




Way prediction set them before tag comparison is done.

# Opt 2 – Way Prediction



# Opt 2 – Way Prediction

- Block associated with prediction (low order tag) bits.
  - → Predicts the next block to be accessed – what locality?
  - → Multiplexer could be set early to select the predicted block, only a single tag comparison
  - → A miss results in checking the other blocks
- Prediction accuracy
  - → > 90% for two-way
  - → > 80% for four-way
  - → I-cache has better accuracy than D-cache
  - → First used on MIPS R10000 in mid-90s
- Extend to predict block as *way selection*
  - → Intends to save power consumption.
  - → Increases mis-prediction penalty

# Opt 3 – Pipelining Cache

- Pipeline cache access to improve bandwidth
  - → Faster clock cycle, but slow hit time
  - → Examples:
    - Pentium: 1 cycle
    - Pentium Pro – Pentium III: 2 cycles
    - Pentium 4 – Core i7: 4 cycles

- Increases branch mis-prediction penalty
- Makes it easier to increase associativity
  - → In associative cache, tag compare and data output are serialized