Cache performance and associativity

Write through vs write back

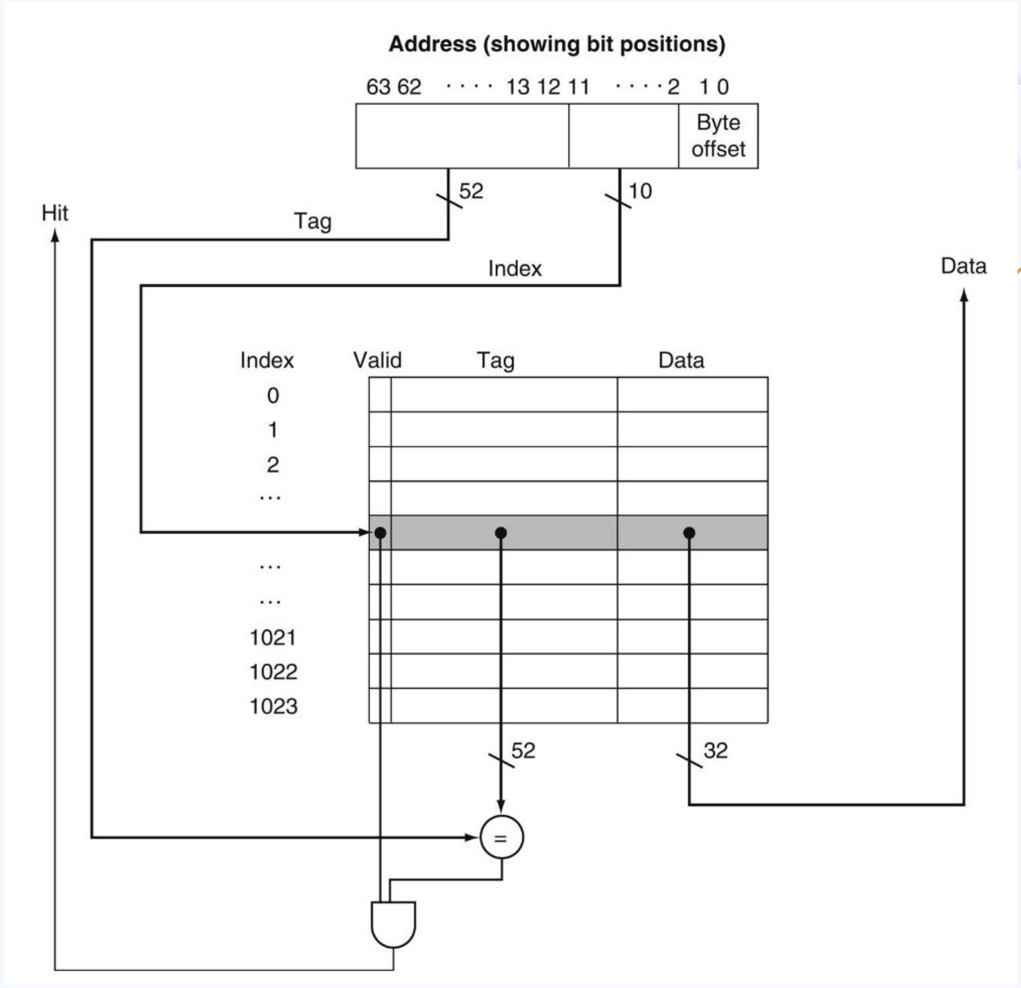
Write through: every time data is changed in cache, change is done to lower level in hierarchy

- Pro: Data are kept consistent (and don't have to write on evict)
- Con: Seems slower (potentially lots of writes to main mem)

Write back: changes to lower level in hierarchy are only done when data is evicted from cache

- Pro: Potentially fewer writes to main memory
- Con: Consistency/complexity issues

Cache controller



P&H fig. 5.10

? ? ?

What happens to the pipeline when a cache miss occurs?

Write buffers

A way to hide the cost of writing to lower level of hierarchy

Example: instruction `sw t1, 4(a0)` in write-through cache

1. Write to cache and write to write buffer happen immediately (simultaneously, 1 cycle)
2. Rest of execution can happen at the same time that write to main memory is happening from buffer

What happens if data that has been evicted from the cache is waiting in the write buffer and a read instruction for that address executes?

? ? ?

How many physical bits of space do we need to store our 1KB cache?

? ? ?

How do we measure the performance of a processor that uses caching?

Formulas from P&H 4.3

$$\text{CPU time} = (\text{CPU execution clock cycles} + \text{Memory-stall clock cycles}) \times \text{Clock cycle time}$$

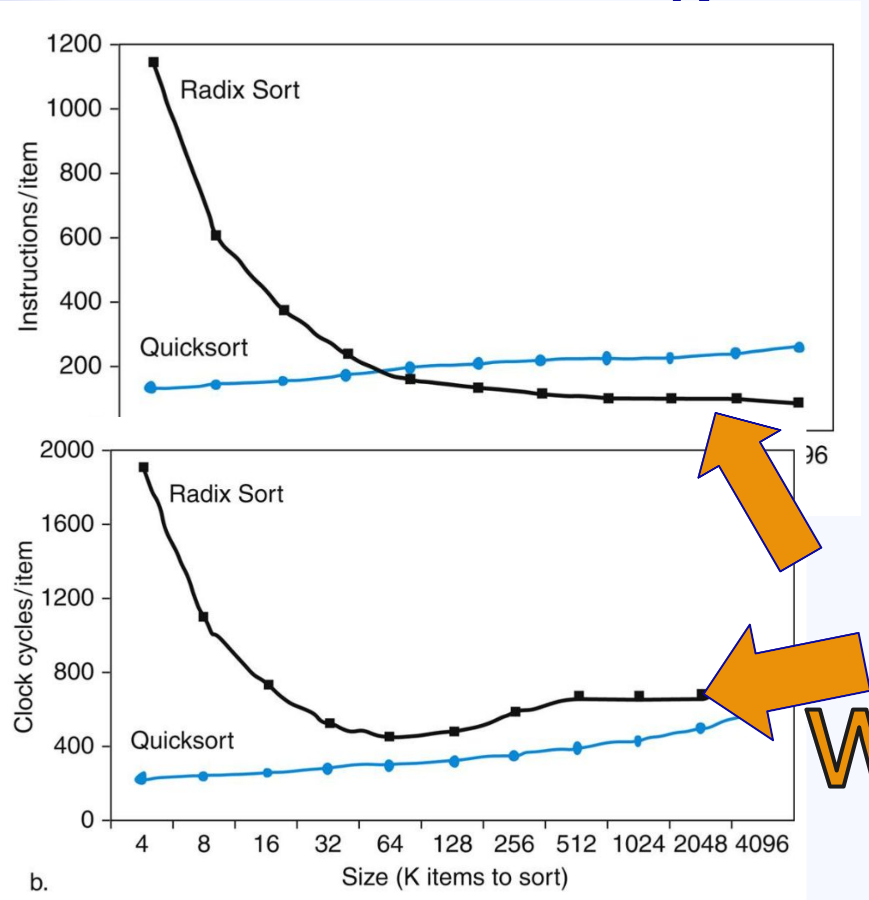
$$\text{Memory-stall clock cycles} = (\text{Read-stall cycles} + \text{Write-stall cycles})$$

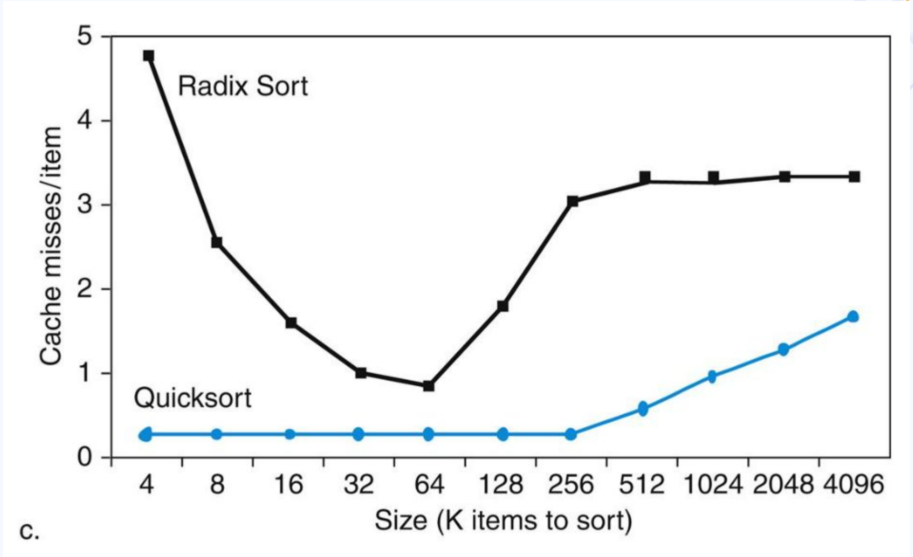
$$\text{Read-stall cycles} = \frac{\text{Reads}}{\text{Program}} \times \text{Read miss rate} \times \text{Read miss penalty}$$

$$\text{Write-stall cycles} = \left(\frac{\text{Writes}}{\text{Program}} \times \text{Write miss rate} \times \text{Write miss penalty}\right) + \text{Write buffer stalls}$$

$$\text{Memory-stall clock cycles} = \frac{\text{Memory accesses}}{\text{Program}} \times \text{Miss rate} \times \text{Miss penalty}$$

Effect of algorithm on CPU time

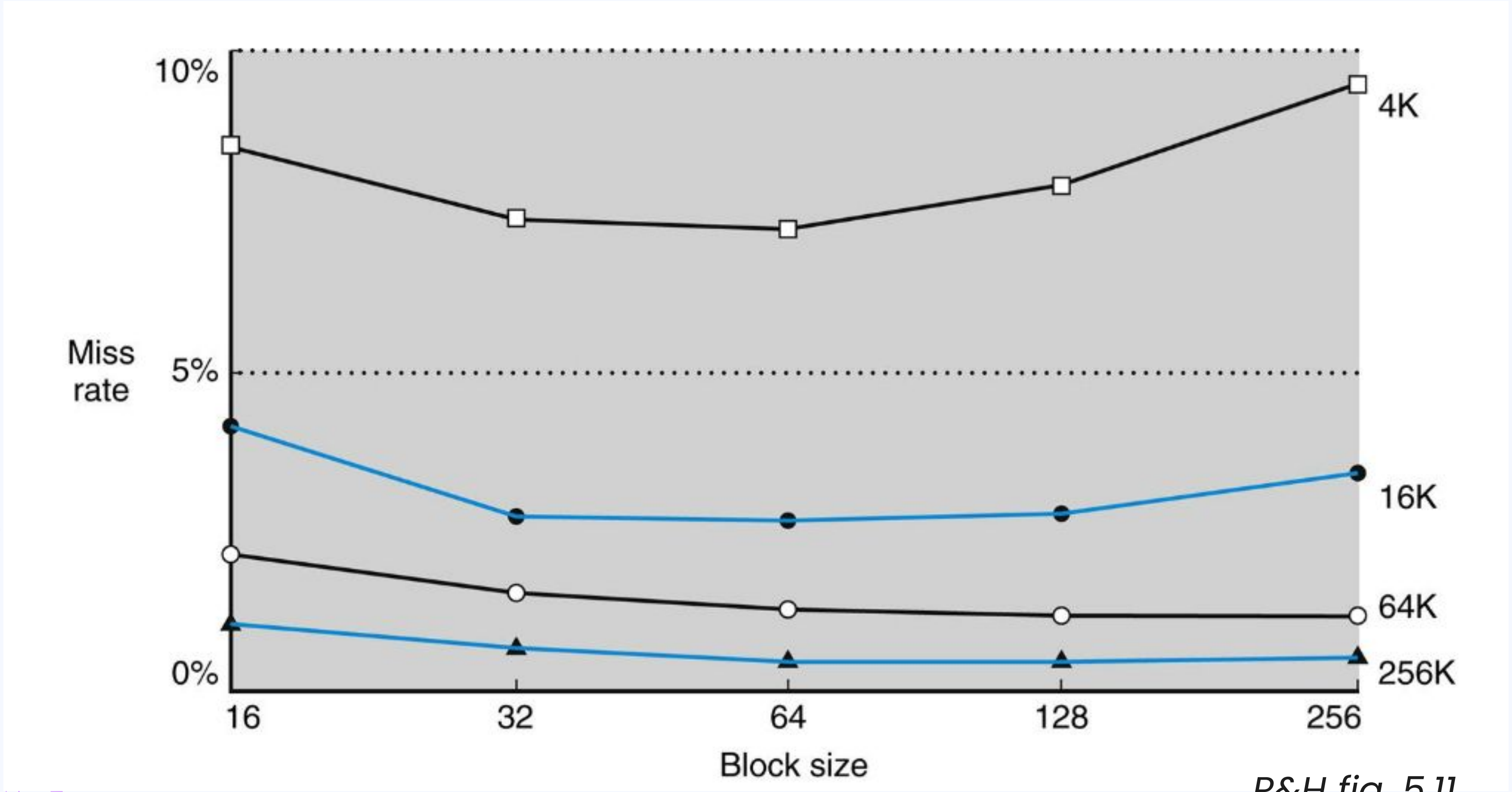




why??

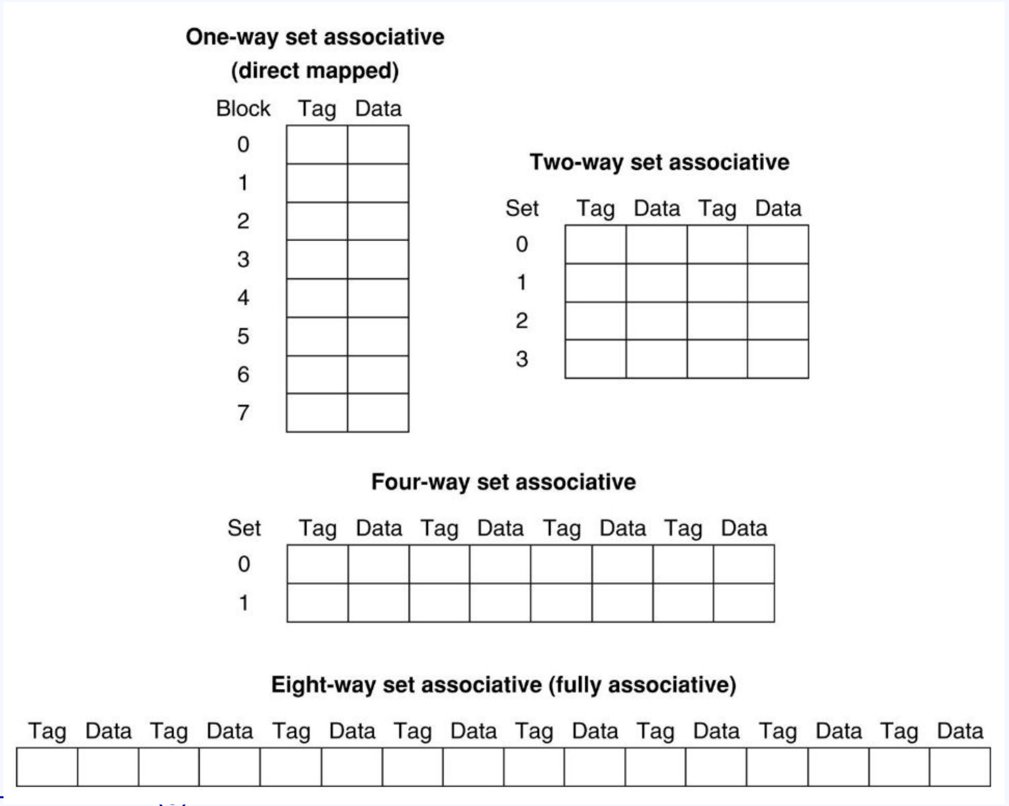
P&H fig. 5.19

Increasing block size has limited effects



P&H fig. 5.11

Set-associative caches


One-way set associative
(direct mapped)
Block Tag Data
0
1
2
3
4
5
6
7
Two-way set associative
Set Tag Data Tag Data
0
1
2
3
Four-way set associative
Set Tag Data Tag Data Tag Data Tag Data
0
1
Eight-way set associative (fully associative)
Tag Data Tag Data Tag Data Tag Data Tag Data Tag Data Tag Data Tag Data

P&H fig. 5.15