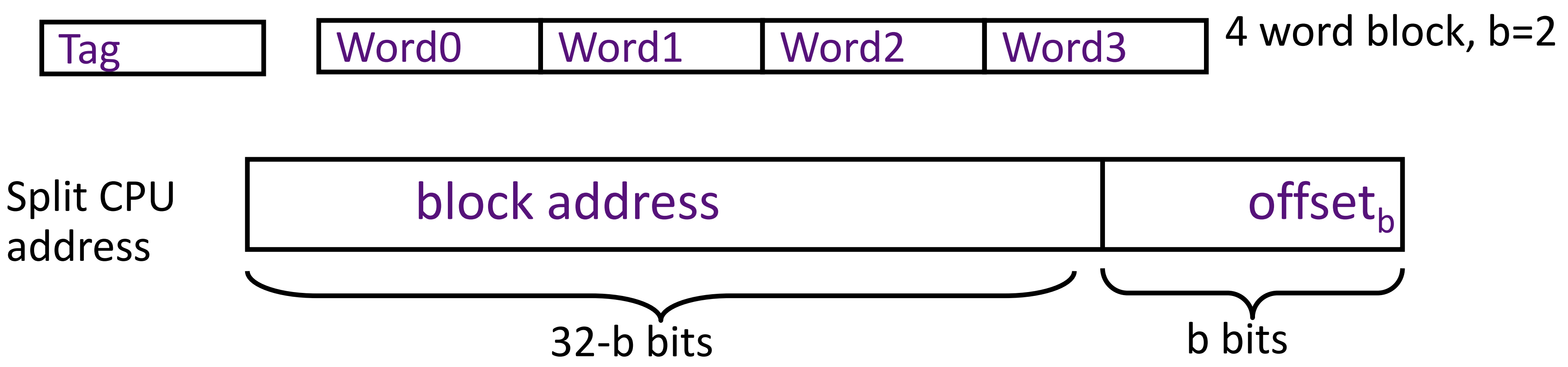
Lecture-15 (Caches)
CS422-Spring 2018

Biswa@CSE-IITK

Block (line) Size ?



2^b = block size *a.k.a* line size (in bytes)

Larger block size has distinct hardware advantages

- less tag overhead
- exploit fast burst transfers from DRAM
- exploit fast burst transfers over wide busses

What are the disadvantages of increasing block size?

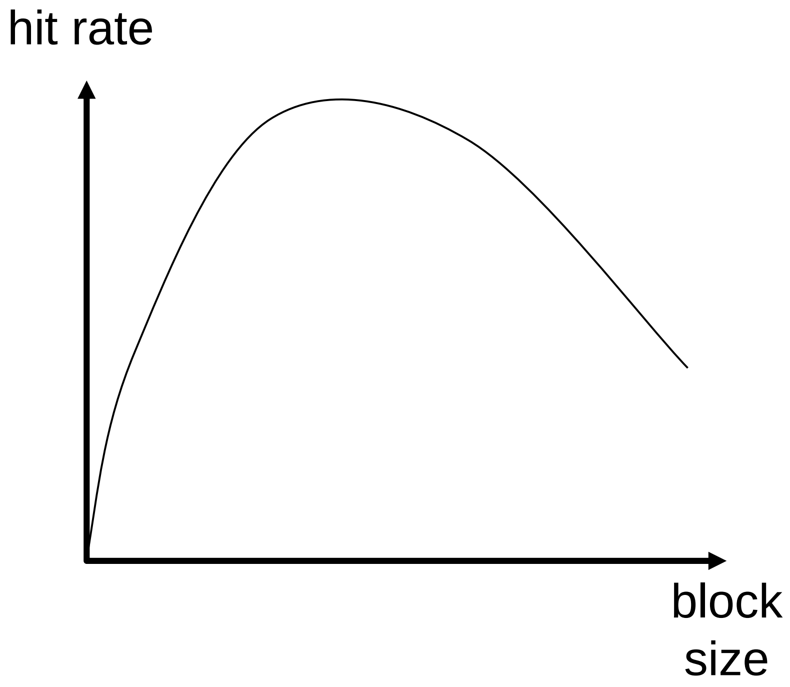
Fewer blocks => more conflicts. Can waste bandwidth.

Block Size?

- Block size is the data that is associated with an address tag
  - not necessarily the unit of transfer between hierarchies
    - Sub-blocking: A block divided into multiple pieces (each with V bit)
      - Can improve "write" performance

- Too small blocks
  - don't exploit spatial locality well
  - have larger tag overhead

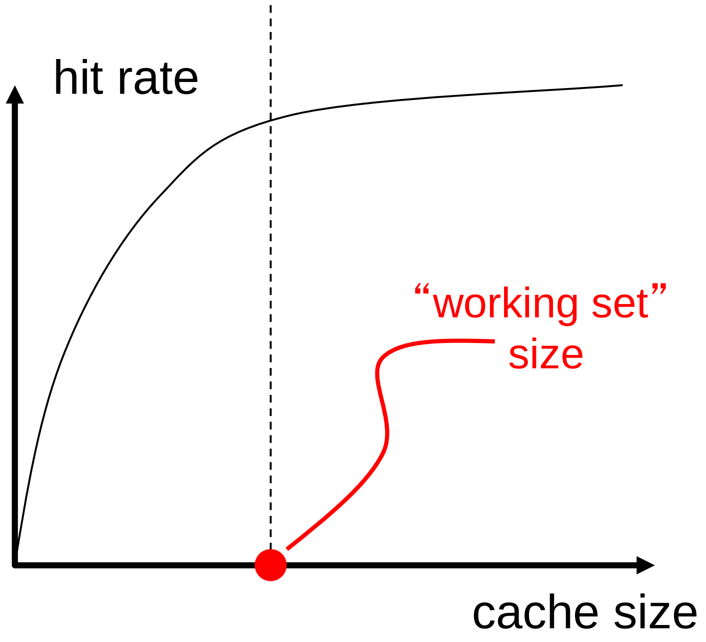
- Too large blocks
  - too few total # of blocks
    - likely-useless data transferred
    - Extra bandwidth/energy consumed



Cache Size

- Cache size: total data (not including tag) capacity
  - bigger can exploit temporal locality better
  - not ALWAYS better
- Too large a cache adversely affects hit and miss latency
  - smaller is faster => bigger is slower
  - access time may degrade critical path
- Too small a cache
  - doesn't exploit temporal locality well
  - useful data replaced often

- Working set: the whole set of data the executing application references
  - Within a time interval



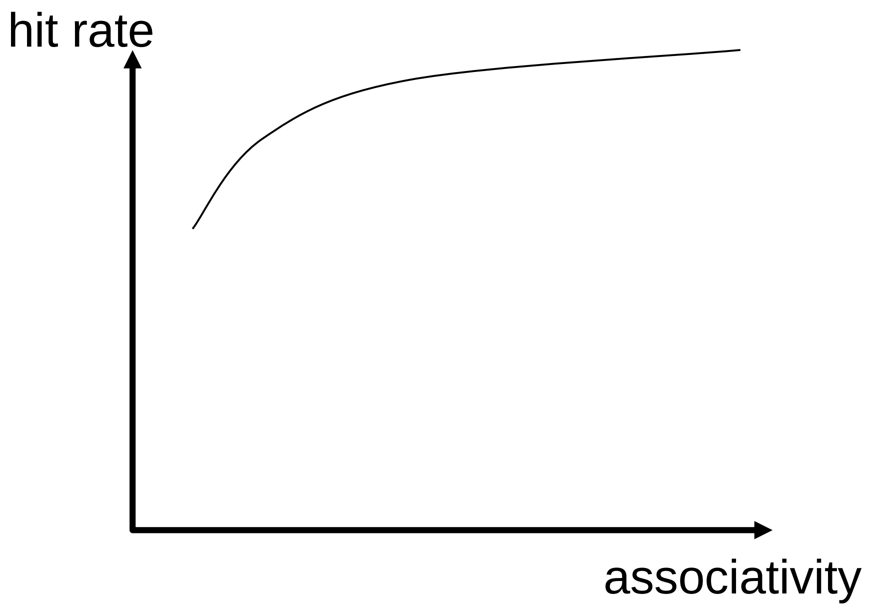
Associativity

- How many blocks can map to the same index (or set)?

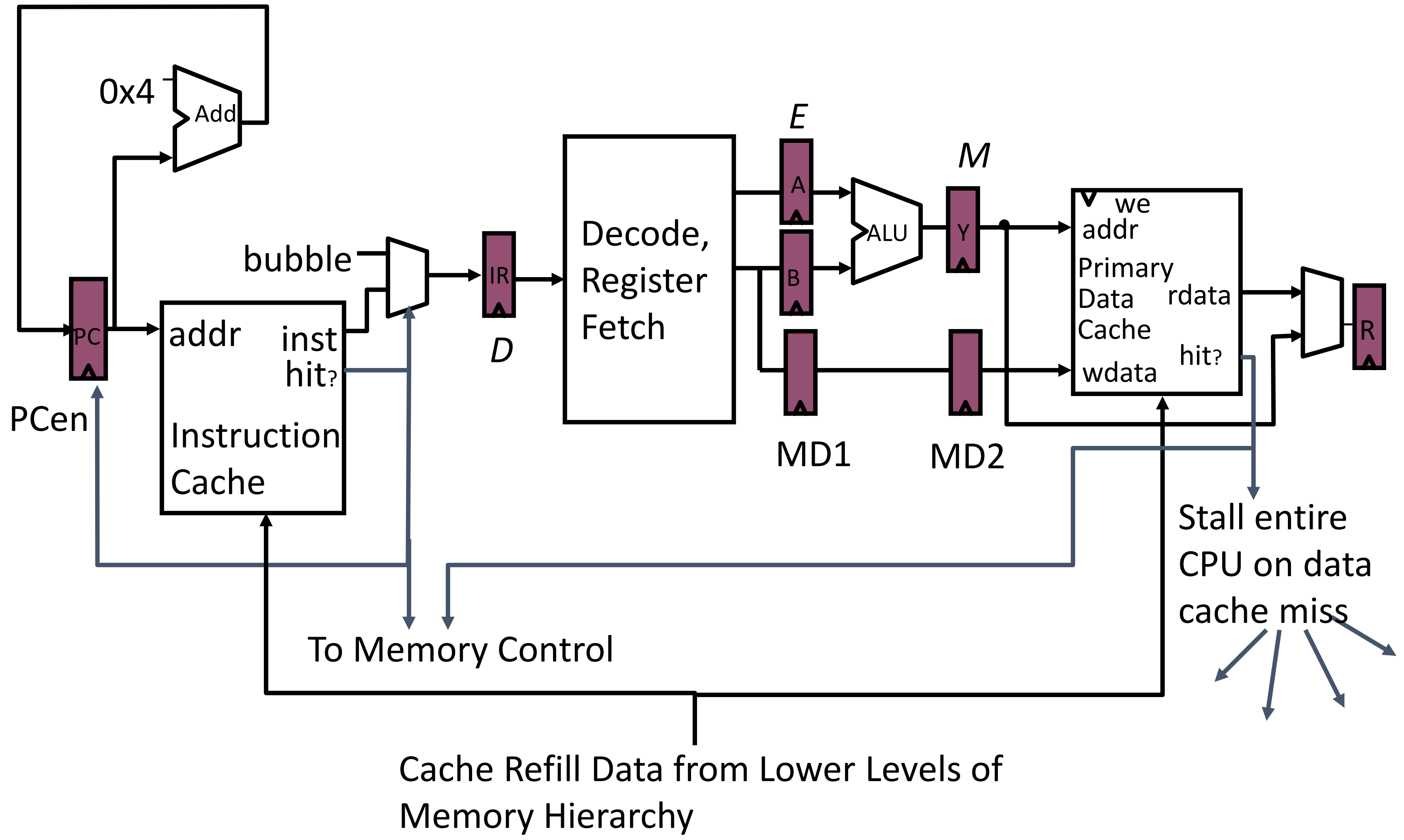
- Larger associativity
  - lower miss rate, less variation among programs
  - diminishing returns, higher hit latency

- Smaller associativity
  - lower cost
  - lower hit latency
    - Especially important for L1 caches

- Power of 2 associativity?



CPU – Cache Interaction



Performance: AMAT

Average memory access time (AMAT) = Hit time + Miss rate x Miss penalty

Average memory access time (AMAT) = Hit time + Miss rate$_1$ x Miss penalty$_1$
+ Miss rate$_2$ x Miss penalty$_2$

Improving Cache Performance

Average memory access time (AMAT) =

Hit time + Miss rate x Miss penalty

To improve performance:

- reduce the hit time
- reduce the miss rate
- reduce the miss penalty

Biggest cache that doesn’t increase hit time past 1 cycle (approx 8-32KB in modern technology)

[design issues more complex with deeper pipelines and/or out-of-order superscalar processors]

The 3Cs

Compulsory:

first reference to a line (a.k.a. cold start misses)

- *misses that would occur even with infinite cache*

Capacity:

cache is too small to hold all data needed by the program

- *misses that would occur even under perfect replacement policy*

Conflict:

misses that occur because of collisions due to line-placement strategy

- *misses that would not occur with ideal full associativity*

Cache Knobs and Performance

- Larger cache size
  - + reduces capacity and conflict misses
  - - hit time will increase
- Higher associativity
  - + reduces conflict misses
  - - may increase hit time
- Larger line size
  - + reduces compulsory and capacity (reload) misses
  - - increases conflict misses and miss penalty