Locality

- Having a memory hierarchy makes sense because of reference locality.
  - If an item is referenced,
    - it will tend to be referenced again soon (temporal locality)
    - nearby items will tend to be referenced soon (spatial locality)

Examples of locality in code and data?

- Our initial focus: two levels (upper, lower)
  - block: minimum unit of data transferred between levels
  - hit: data requested is in the upper level
  - miss: data requested is not in the upper level
  - hit rate: percentage of hits (sometimes called hit ratio)
  - miss rate: percentage of miss (sometimes called miss ratio)
  - hit time: the time to satisfy request in case of a hit
  - miss penalty: the time to satisfy a request in case of a miss
The Basics of Caches

- **First assumption**: block is one word of data
- **Three issues**:
  - How do we know if a data item is in the cache?
  - If it is, how do we find it?
  - If it is not, what do we do?
- It boils down to
  - where do we put an item in the cache?
  - how do we identify the items in the cache?
- **Two solutions**
  - put item anywhere in cache (associative cache)
  - associate specific locations to specific items (direct mapped cache)

Direct Mapped Cache

- **Address is modulo the number of blocks in the cache**

For each item of data in memory, there is exactly one location in the cache where it might be.

\[
\text{Index} = (\text{memory address}) \mod (\text{cache size})
\]

Location in cache

**Note**: lots of items in memory share same location in cache → need to tag each item

- Cache is accessed using the low-order address bits (hashing)
- Tag data using the high order address bits
- A valid bit to detect when location does not contain valid data
- Ex: 00110, 11010, 10110, 10000, 00011, 10000, 10010

Same index → collision → tag is used for differentiation
Direct Mapped Cache (example of a 4KB cache)

- What is the total number of bits needed for a 4KB cache assuming 32-bit address?
- What about an 8KB cache?

Hits vs. Misses

- Read hits:
  - this is what we want!

- Read misses:
  - stall the CPU, fetch block from memory, deliver to cache, restart the read
  - may have to replace (overwrite) data already in cache
  - need a replacement algorithm??

- Write hits:
  - write data to cache. May (write through) or may not (write back) write data also to memory. Will talk about this later.

- Write miss:
  - write data to cache. May or may not write data also to memory
  - may have to replace (overwrite) data already in cache.
  - what do you do with the data that you have to replace??