- Accessing mem is slow:
  - May rely on delay for communication (no mem, system, selecting, links, routing)
  - R/W bits is fast
    "Memory Wall"

- Register access: ~5 ns
- SRAM access: ~1-5 ns
- DRAM access: ~50-70 ns
  ~$10
- Disk (not SSD): 5,000,000 -> 20,000,000 ns
  5ms -> 20ms

- We want to take advantage of fast, small memory technologies. But how?
- Spatial Locality: we tend to access things close together in memory
- Temporal Locality: if we access something, chances are we'll access it again soon
- Research at a library analogy
- **Memory Hierarchy**

- We move stuff from one level to another at a time.
- We move stuff in units of blocks or lines.
- Present in closest level? **"Hit"**
  Not? **"Miss"**
- Hit Rate: Ratio of hits to total # of accesses
- Hit Time: How long a hit takes
- Miss Penalty: How long it takes to access lower level and bring it to CPU.
- Our higher levels we call a **cache**
- Generally refers to storage that takes advantage of locality of reference.
Anything \% \#ENTRIES == 0 goes here

We use addresses to index the cache.

How do we know which one is there? Add a tag.

- How do we add lines > 1 block?
- In practice, cache line sizes are ~64 B
  (32 LC-3 words)

- Measuring memory performance:

  Average memory access time (AMAT) -

  \[ \text{AMAT} = t_h + P_m R_m \]

  \( t_h \) - hit time
  \( P_m \) - miss penalty
  \( R_m \) - miss rate
- 2 options:
  1) Improve Miss Rate
  2) Reduce Miss Penalty

2) Add more levels of memory!

1) Write Better Software (hard) \rightarrow \text{array stride}
   
   Design better cache (easy) \rightarrow \text{set associativity}
   
   Prefetching

Blocks (Lines) can map to more than one location.
- More expensive, must add comparators, check all "sets" in parallel

When we kick someone out, how do we pick?
- Pick oldest \underline{must be} most stale element. "Least Recently Used" (LRU).
- Add counters to cache lines for example
HANDLING WRITES

- WRITE THROUGH: PROPAGATE WRITES DOWN TO MAIN MEM.

  EXPENSIVE! DEFENDS PURPOSE

- WHY NOT ADD A WRITE BUFFER?

  IF CPU GENERATES WRITES FASTER, WILL ALWAYS FILL UP. NO BETTER!

- WRITE BACK: ONLY PROPAGATE WRITES WHEN EVICTED, CACHES ARE KEPT INCONSISTENT WITH MAIN MEMORY