ARCHITECTURE I: MEMORY HIERARCHY

- ACCESSING MEM IS SLOW:
  - MYTH: FROM DELAY for COMMUNICATION W/ MEM. SYSTEM, SELECTING LINES, ROUTING
  - R/W BITS IS FAST

"MEMORY WALL"

REGISTER ACCESS: ~.5 ns — $/GB $4k - $10k

SRAM ACCESS: ~1-5 ns

DRAM ACCESS: ~50 - 70 ns — ~$10

DISK (NOT SSD): 5,000,000 → 20,000,000 ns — ~$.02
5ms → 20ms

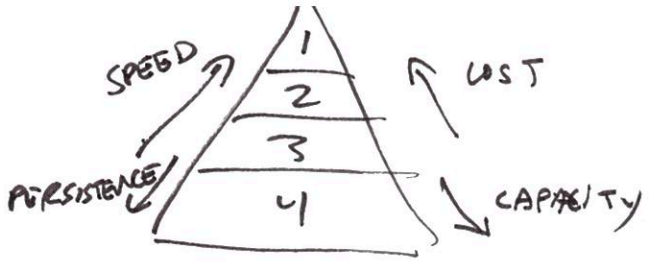
- WE WANT TO TAKE ADVANTAGE OF FAST, SMALL MEMORY TECHNOLOGIES. BUT HOW?

- 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"

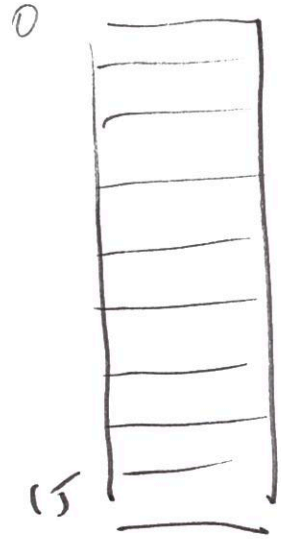
- HIT RATE: RATIO OF HITS TO TOTAL # OF ACCESSES

- 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

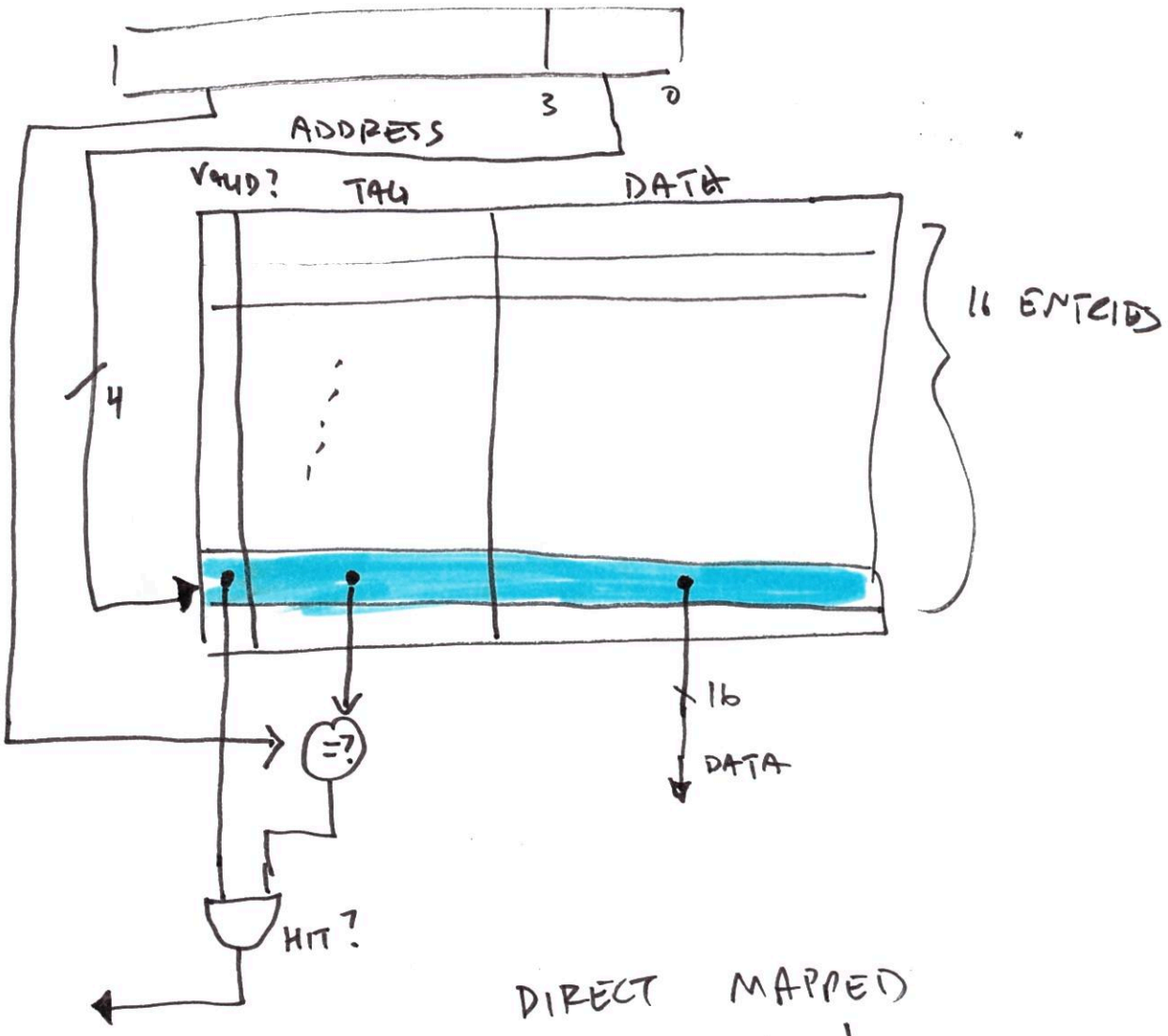
- GENERALLY REFERS TO STORAGE THAT TAKES ADVANTAGE OF LOCALITY OF REFERENCE



ANYTHING % #ENTRIES == 0 GOES HERE

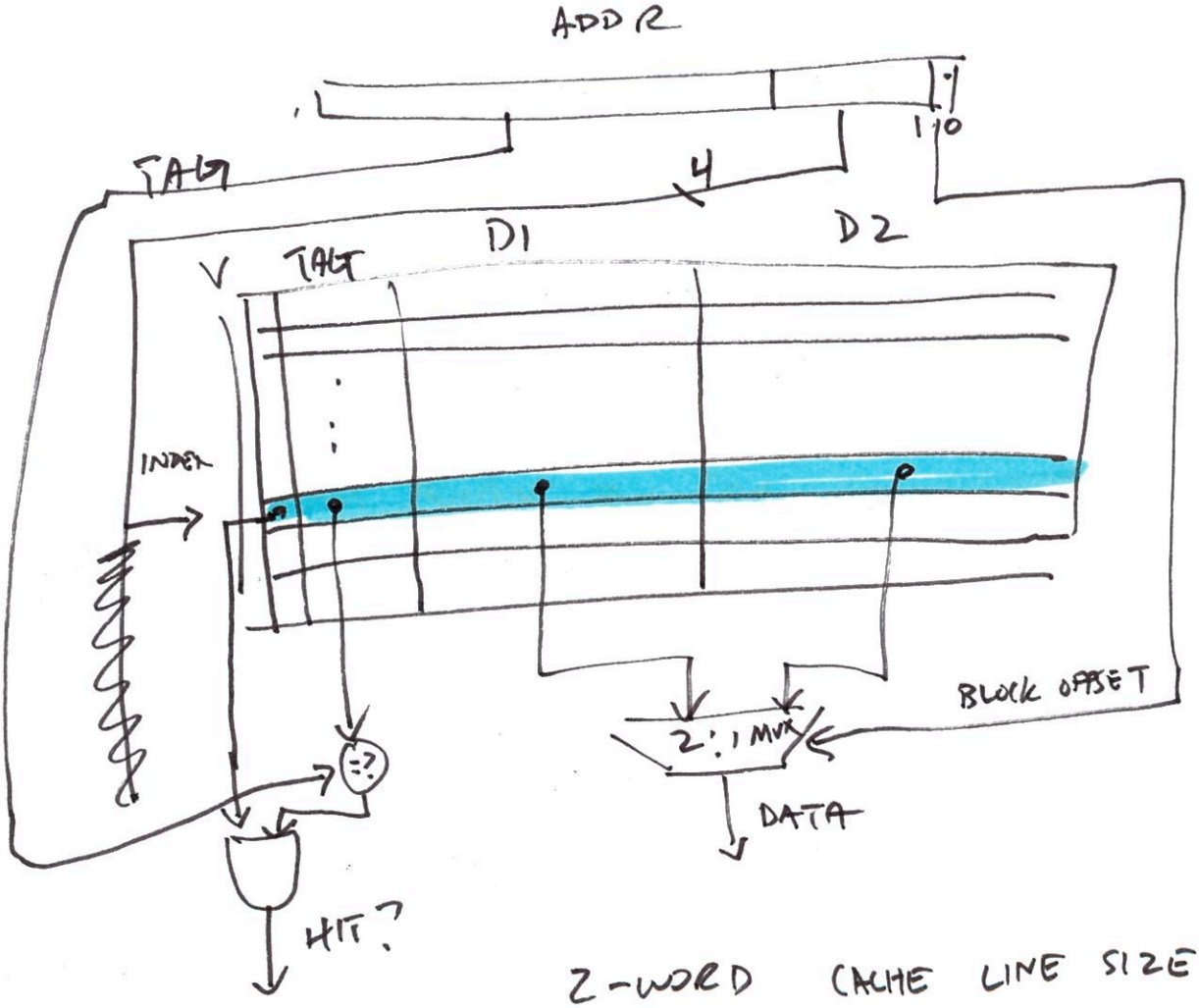
WE USE ADDRESSES TO INDEX THE CACHE

HOW DO WE KNOW WHICH ONE IS THERE? ADD A TAG



DIRECT MAPPED CACHE!

- HOW DO WE ADD LINES > 1 BLOCK?



2-WORD CACHE LINE SIZE

- IN PRACTICE, CACHE LINE SIZES ARE ~64 B (~ 32 LC-3 WORDS)

- MEASURING MEMORY PERFORMANCE:

AVERAGE MEMORY ACCESS TIME (AMAT) -

HIT TIME = t_h
MISS PENALTY = P_m
MISS RATE = R_m

$$AMAT = t_h + P_m R_m$$

- 2 options:

1) IMPROVE MISS RATE
2) REDUCE MISS PENALTY

2) ADD MORE LEVELS OF MEMORY!

1).
- WRITE BETTER SOFTWARE (HARD) → E.G. ARRAY STRIDE
- DESIGN BETTER CACHE (EASY) → 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 OR 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! DEFEATS 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