How do we build a cache?

Cache Functionality

- CPU gives cache full address to access data
- cache has small set of values from main memory
- Fully Associative Cache: set of (Address, Data)'s
  (key, value)
  - You lookup address, return data
    - ☹ cache stores BOTH
  - If (address = found): cache HIT
    - no need to access main memory
    - else, cache MISS ☹ have to access

Locality

- Temporal locality: using X? will need again soon!
  - LOOPS
- Spatial locality: using stuff in address X? why not (X±1, 2....)?
  - instructions, ARRAYS
  - ☹ Bring in ≥1 pieces of data

Cache Hit rate

- % of cache hits
  - CHR for instructions better than those for data
    - ⊕ more regular data patterns
    - ⊕ more locality

How to deal with Cache Misses

- "Put recently used data in cache" – Temp. locality

Cache replacement policy:

- LRU: Least Recently Used (memory access)
  - makes sense
  - expensive to implement

- Round robin:
  - cycle round locations
  - Least recently fetched from memory
- Random: $(\frac{1}{N})$ chance of an address being evicted
  - easy to implement

Cache WRITING

- Writes more difficult than reads
- Have to compare addresses to check if address in cache
  - Hit write strategy:
    - Write Through
    - ???, with buffers
    - Copy Back
  - Miss write strategy:
    - Write Allocation
      1) Find location
         -
      2) Assign cache location, write value
      3) Write through back to RAM

Cache Data Width

- Line
  - | Address | Data |

Real Cache Implementation

- We always need a full associative cache
  - (!) Expensive (!)

Direct Mapped Cache

- Address is 2 parts: Tag, Index
  - Hash Map implementation
- RAM divided into blocks the SIZE of the cache
  - each mem. location has assigned cache entry
  - Many RAM locations compete for same cache locations
- A HASH TABLE implemented in hardware
  - many ways to select Index & Tag from address

Set Associative caches:

- Set of DMC's operating in parallel
- replacement strategy way more flexible
  - e.g. in 4-way SA cache, any cache choosable for cache replacement
- Hit rate is MUCH better (more ways to cache results)
  - $$$ obviously

	Cache 0	Cache 1
00	A	B
01	C	D
10	E	F
11	G	H

☹ fetching data from address 11010

Block at LSB's 10 ☹

if Random, E or F,