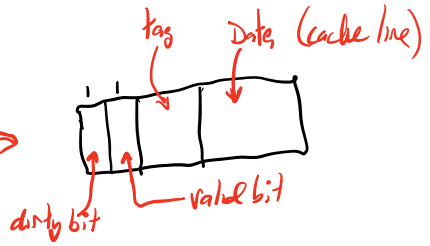
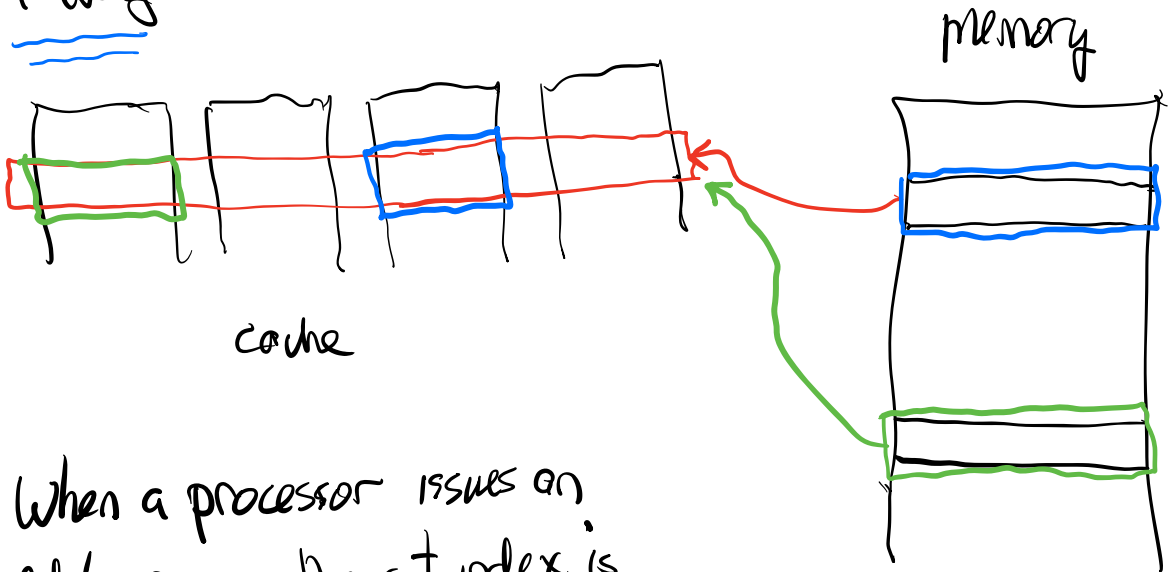
# Set-Associative Cache

Each cache entry in a set looks like →




4-way:




When a processor issues an address, once the set index is computed, the cache hardware has to look at every entry in that set for v=1 and a matching tag.

- all entries in the set are checked simultaneously for v=1 and a tag that matches the tag bits in the address.
  - if an entry satisfies these requirements, it's a cache hit.

"Fully Associative Cache"

- a cache line in memory can be put into any entry of the cache.
- this is an n-way set associative cache where n is equal to the total number of cache entries in the cache

$$n = N/M$$

- when a processor issues an address, every entry in the cache has to be checked for a cache hit.
  - too expensive (not used)

# Cache Replacement Policy

- which entry in a set should we choose to put an incoming cache line into?
- may have to evict a cache line that is already in the chosen entry.
- if so, we want to evict a cache line that we don't need any time soon.

Possible policies:

1. Random
   - randomly choose an entry in the set to place the new cache line in.
   - Might be evicting a cache line that we need.

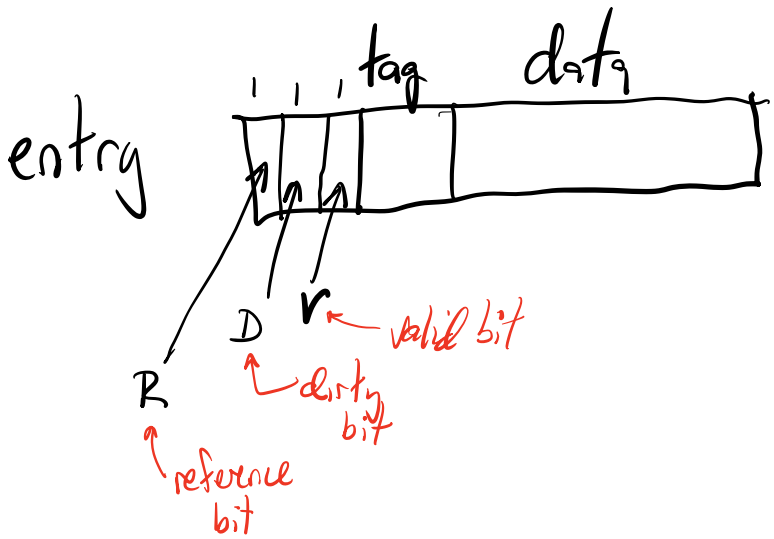
2. Least recently Used (LRU)
   - takes advantage of the property that if a memory location hasn't been used recently, then it probably won't be used again any time soon.

- LRU chooses to evict the cache line that hasn't been used (referenced) in the longest time.

- the hardware has to keep track of the last time each entry in a set was referenced.
  - expensive (have to store a time stamp)
  - not used in actual cache hardware

# (3) Not Recently Used (NRU)

- approximation to LRU
- keeps track of which cache entries were used (referenced) recently.
  - doesn't keep track of when they were used, just whether or not they were used recently.
- picks a cache entry to evict that (hopefully) wasn't used recently.
- uses one additional bit in each cache entry: Reference Bit (R)



- periodically (e.g. every 20 ms), all the R bits in the cache are set to 0.
- every time a cache entry is accessed by a read or write, the R bit is set to 1.

So, those cache entries with R=1 have been used recently.

When bringing a new cache line into the cache, we choose an entry in the set according to the following priority:

1. V=0 - there's no data that has to be evicted.

2. R=0 D=0 - evict a cache entry that hasn't been used recently and doesn't need to be written back.

3. R=0 D=1 - evict an entry that hasn't bee used recently, but has to be written back to memory.

4. R=1 D=0 - will likely need this cache line soon, but at least we don't have to write it back to memory.

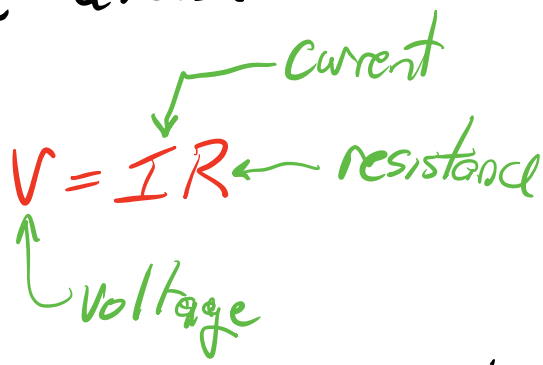
5. R=1 D=1 - worst case, just pick one

This is what caches actually use (NRU).

# Digital Logic

- designing the circuits of processors, memories, etc.

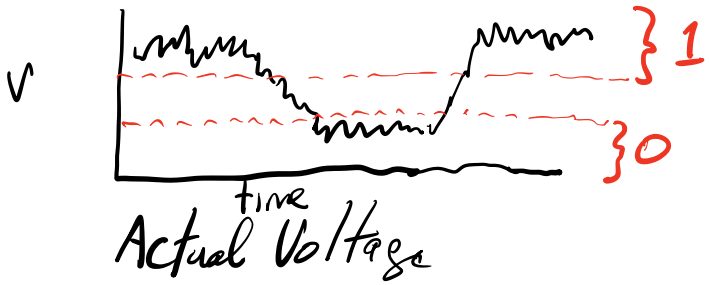
- 0's and 1's are represented by voltage levels.




  - given a fixed resistance, voltage and current are proportional.

0 is represented by low voltage
  - low amount of current (electrons) on a wire.

1 is represented by high voltage.



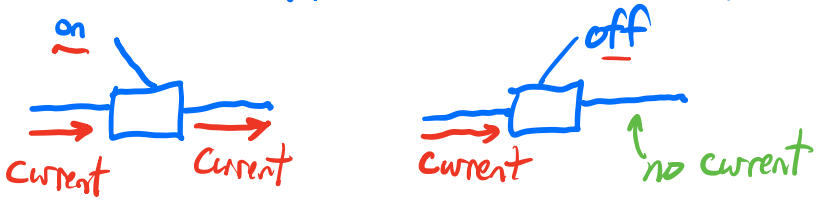


this is how we think about voltage levels.

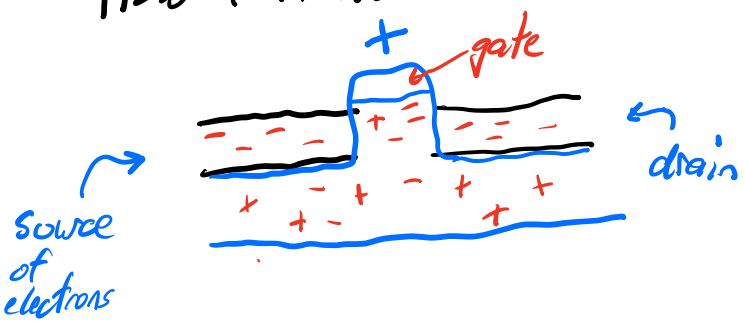
Modern circuits are built from transistors.

transistor = "transmitter" and "resister"

- think of a transistor as a light switch:




- How a transistor works:




When a positive charge is exerted at the gate, the electrons flow from the source to the drain.