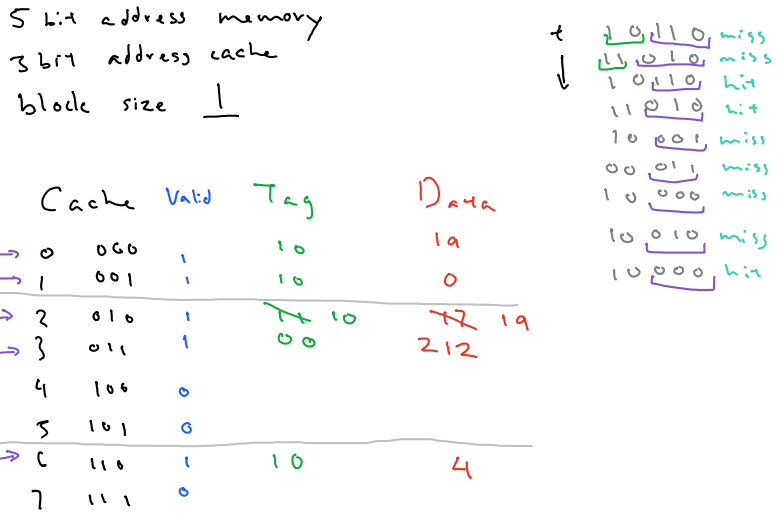
25 Direct Mapped Cache

Monday, November 2, 2015 10:02 AM



MISS

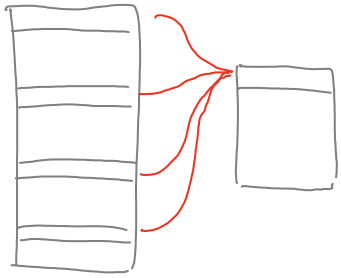
1. Grab last three bits, to get index in cache
2. Check valid bit
    a. 0 so a miss, go to memory to find data
3. Put the data from memory into the data in cache
4. Put the first two bits in tag
5. Toggle valid bit

HIT

1. Grab last three bits, to get index in cache
2. Check valid bit
    a. 1 so a hit
3. Check tag, see if same
4. Return data stored in cache

Direct mapped cache

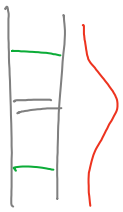
- Any RAM address has one spot in cache
- Any cache address has n (=4) different RAM addresses



The way this is arranged affects, the miss/hit rate for consecutive polls

0	00000
1	00001
2	00010
3	00011
4	00100
5	00101
6	00110
7	00111
8	01000
9	01001
10	01010
11	01011
12	01100
13	01101
14	01110
15	01111
16	10000
17	10001
18	10010
19	10011
20	10100
21	10101
22	10110
23	10111
24	11000
25	11001
26	11010
27	11011
28	11100
29	11101
30	11110
31	11111

BLOCK SIZE



- Amount of data RAM <-> Cache
- Spatial locality suggests
    - No perfect block size, but should be larger than one byte or one word
- Block size ++ (what if we increase block size)
    - But there are consequences
        - You need more cache memory
        - More memory per index in cache
        - Thus my cache is $$
- If we keep memory constant -> fewer slots
    - Fewer slot -> more collisions/evictions
- BS++, Miss penalty also ++
    - Need to transfer more stuff from RAM to cache

Not just temporal locality
Also spatial locality, yields block size

BURST MODE transfer (RAM technology)

- Slower to access two indexes that are far apart
- Faster to access one larger block that is consecutive

How to transfer a block