Caches

SEB

$S = 2^s$ sets
$E =$ lines per set
$B = 2^b$ bytes per cache block (excluding valid & tag)

Cache size = $S \cdot E \cdot B$ data bytes

3 parts

t bits	s bits	b bits

tag | set index | block offset

1. Locate set. Which set in cache does address match to?
2. Check if any line in set has matching tag
3. Yes + line valid : hit
4. Locate data @ data offset

valid bit : is it valid or not

direct mapped cache: E = 1 , 1 line

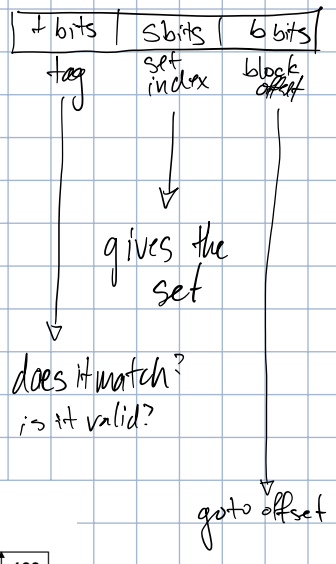
assume block size is 8 bytes : B = 8 b = 3

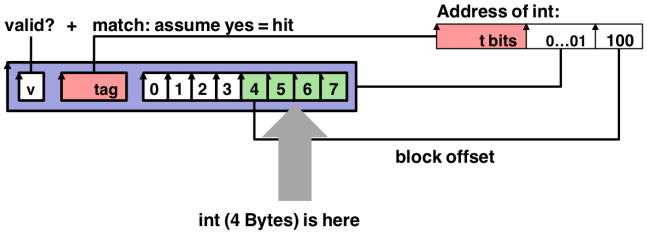
address of word (8 bytes):

t bits	s bits	b bits

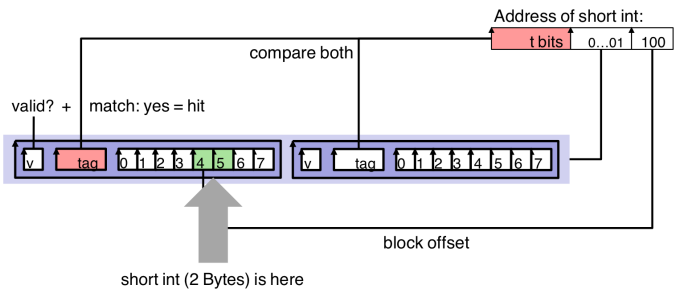
tag | set index | block offset

4 sets : s = 2





2 way set associative: E=2 2 blocks per set



No tag hit? → miss → go to memory

direct: 1 block to 1 set
2 way: 2 blocks to 1 set
fully associative: only 1 set

Which block from set should be evicted when set is full?

① Random

② Least Recently Used

③ First in First Out

How about writes?

Cache hit

write through – write to memory immediately
write back – write cache only, memory written when evicted

Cache miss

no write allocate – write to main memory
write allocate – fetch block, then write

Cache performance metrics

① Miss rate — L1: 3-10% L2: <1%

② Hit time — time to deliver line in cache to processor
L1: 1-4 clock cycles L2: 10-30 block cycles

③ Miss penalty — additional time required b/c of a miss

97 vs 99 % hit rate

consider: cache hit time of 1 cycle
miss penalty of 100 cycles

average memory access time = hit time + miss rate * miss penalty

97% hits → 1 cycle + 0.03 · 100 cycles = 4 cycles
99% hits → 1 cycle + 0.01 · 100 cycles = 2 cycles

Types of misses

① Compulsory / cold — first time — needs
② Capacity — cache too small — hold all data needed by program — make bigger
③ Conflict — collisions — ???