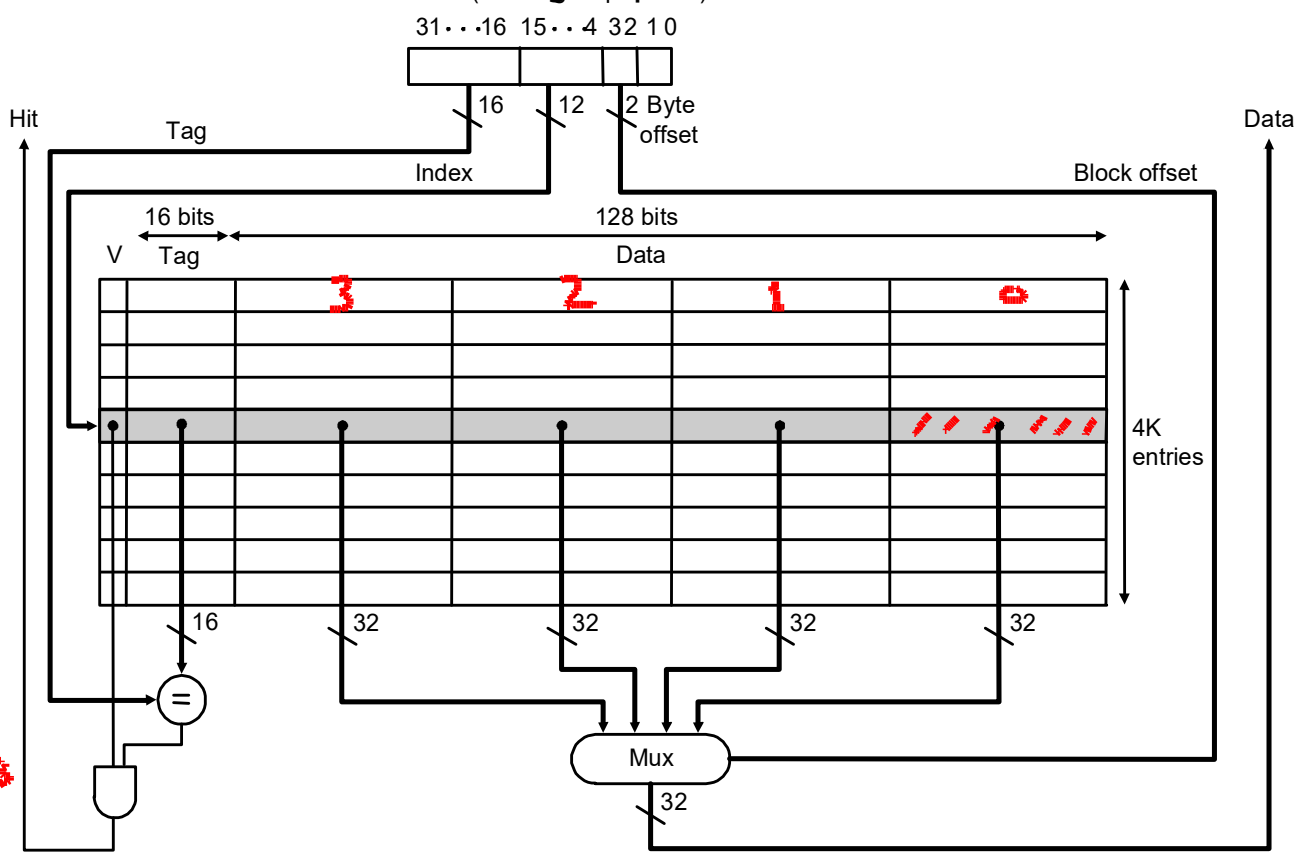
Direct Mapped Cache: Taking Advantage of Spatial Locality

- Taking advantage of spatial locality with *larger* blocks:



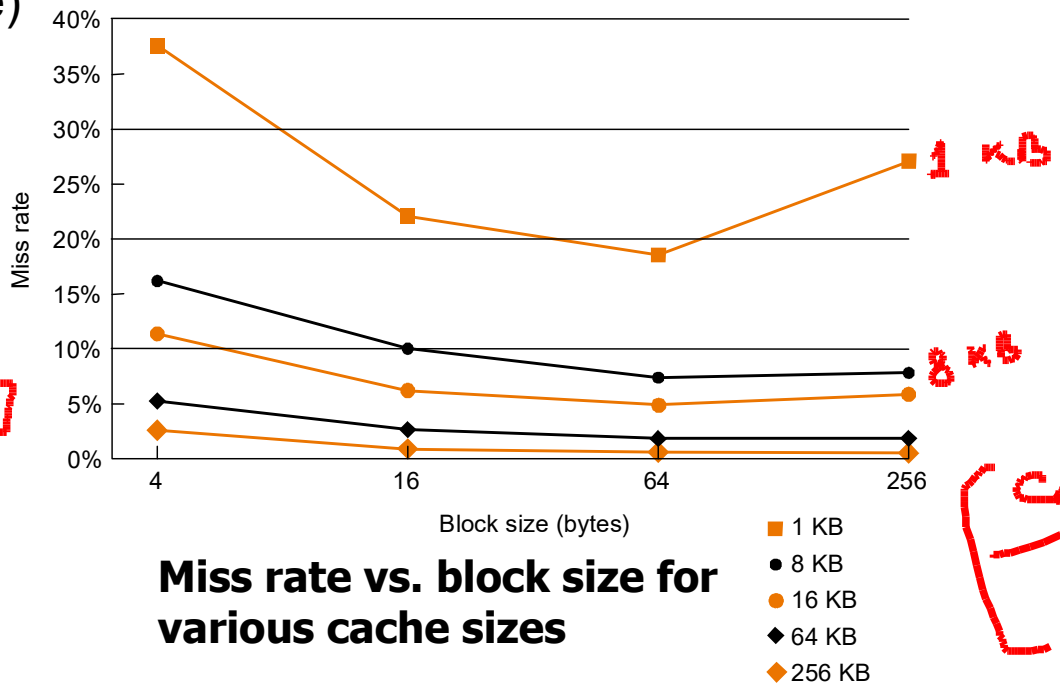
Cache with 4K 4-word blocks: *byte offset* (least 2 significant bits) is ignored, next 2 bits are *block offset*, and the next 12 bits are used to index into cache

Direct Mapped Cache: Taking Advantage of Spatial Locality

- *Cache replacement* in large (multiword) blocks:
  - word *read miss*: read entire block from main memory
  - word *write miss*: *cannot* simply write word and tag! *Why*?!
  - writing in a *write-through* cache:
    - if *write hit*, i.e., tag of requested address and and cache entry are equal, continue as for 1-word blocks by replacing word and writing block to both cache and memory
    - if *write miss*, i.e., tags are unequal, fetch block from memory, replace word that caused miss, and write block to both cache and memory
    - therefore, unlike case of 1-word blocks, a write miss with a multiword block causes a memory read

Direct Mapped Cache: Taking Advantage of Spatial Locality

- Miss rate falls at first with increasing block size as expected, but, as block size becomes a large fraction of total cache size, miss rate may go up because
  - there are few blocks
  - competition for blocks increases
  - blocks get ejected before most of their words are accessed (*thrashing* in cache)



Miss rate vs. block size for various cache sizes

Improving Cache Performance

Use split caches for instruction and data because there is more spatial locality in instruction references:

Ins. cache D. cache

Program	Block size in words	Instruction miss rate	Data miss rate	Effective combined miss rate
gcc	1	6.1%	2.1%	5.4%
	4	2.0%	1.7%	1.9%
spice	1	1.2%	1.3%	1.2%
	4	0.3%	0.6%	0.4%

Miss rates for gcc and spice in a MIPS R2000 with one and four word block sizes

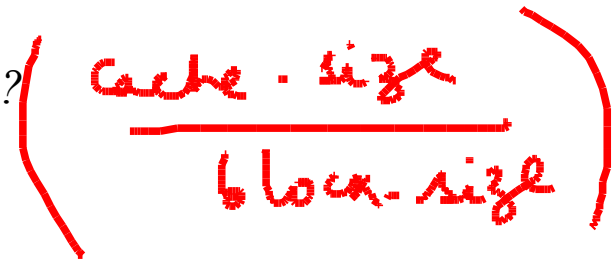
- Make reading multiple words (higher bandwidth) possible by increasing physical or logical width of the system…

block size = unit of data transfer

Performance

- Simplified model assuming equal read and write miss penalties:
  - CPU time = (execution cycles + memory stall cycles) × cycle time
  - memory stall cycles = memory accesses × miss rate × miss penalty

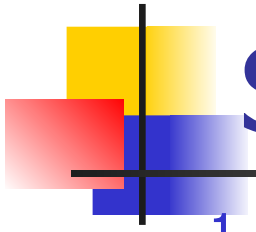
- Therefore, two ways to improve performance in cache:
  - decrease miss rate
  - decrease miss penalty
  - *what happens if we increase block size?*



Example Problems

- Assume for a given machine and program:
  - instruction cache miss rate 2%
  - data cache miss rate 4%
  - miss penalty always 40 cycles
  - CPI of 2 without memory stalls
  - frequency of load/stores 36% of instructions

1. *How much faster is a machine with a perfect cache that never misses?*
2. *What happens if we speed up the machine by reducing its CPI to 1 without changing the clock rate?*
3. *What happens if we speed up the machine by doubling its clock rate, but if the absolute time for a miss penalty remains same?*



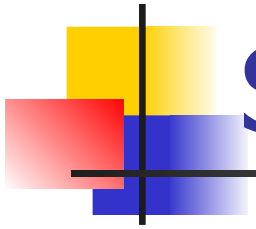
Solution

1.

- Assume instruction count = I
- Instruction miss cycles = I × 2% × 40 = 0.8 × I
- Data miss cycles = I × 36% × 4% × 40 = 0.576 × I
- So, total memory-stall cycles = 0.8 × I + 0.576 × I = 1.376 × I
  - in other words, 1.376 stall cycles per instruction
- Therefore, CPI with memory stalls = 2 + 1.376 = 3.376
- Assuming instruction count and clock rate remain same for a perfect cache and a cache that misses:

  CPU time with stalls / CPU time with perfect cache

  = 3.376 / 2 = 1.688
- Performance with a perfect cache is better by a factor of 1.688



Solution (cont.)

2.

- CPI without stall = 1
- CPI with stall = 1 + 1.376 = 2.376 (clock has not changed so stall cycles per instruction remains same)
- CPU time with stalls / CPU time with perfect cache

  = CPI with stall / CPI without stall

  = 2.376
- Performance with a perfect cache is better by a factor of 2.376
- Conclusion: with higher CPI cache misses “hurt more” than with lower CPI

Solution (cont.)

3.

- With doubled clock rate, miss penalty = 2 × 40 = 80 clock cycles
- Stall cycles per instruction = (I × 2% × 80) + (I × 36% × 4% × 80)
  = 2.752 × I
- So, faster machine with cache miss has CPI = 2 + 2.752 = 4.752
- CPU time with stalls / CPU time with perfect cache
  = CPI with stall / CPI without stall
  = 4.752 / 2 = 2.376
- Performance with a perfect cache is better by a factor of 2.376
- Conclusion: with higher clock rate cache misses "hurt more" than with lower clock rate