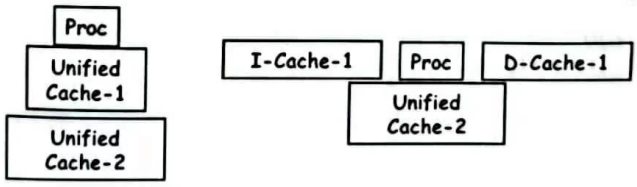
Impact on Performance

- Suppose a processor executes at
  - Clock Rate = 200 MHz (5 ns per cycle), Ideal (no misses) CPI = 1.1
  - 50% arith/logic, 30% ld/st, 20% control
- Suppose that 10% of memory operations get 50 cycle miss penalty
- Suppose that 1% of instructions get same miss penalty
- CPI = ideal CPI + average stalls per instruction
  1.1(cycles/ins) +
  [0.30 (DataMops/ins)
  x 0.10 (miss/DataMop) x 50 (cycle/miss)] +
  [1 (InstMop/ins)
  x 0.01 (miss/InstMop) x 50 (cycle/miss)]
  = (1.1 + 1.5 + .5) cycle/ins = 3.1
- 58% of the time the proc is stalled waiting for memory!
- AMAT=(1/1.3)x[1+0.01x50]+(0.3/1.3)x[1+0.1x50]=2.54

Example: Harvard Architecture

- Unified vs Separate I&D (Harvard)



- Table on page 384:
  - 16KB I&D: Inst miss rate=0.64%, Data miss rate=6.47%
  - 32KB unified: Aggregate miss rate=1.99%
- Which is better (ignore L2 cache)?
  - Assume 33% data ops ⇒ 75% accesses from instructions (1.0/1.33)
  - hit time=1, miss time=50
  - Note that *data* hit has 1 stall for unified cache (only one port)

$AMAT_{Harvard}$=75%x(1+0.64%x50)+25%x(1+6.47%x50) = 2.05
$AMAT_{Unified}$=75%x(1+1.99%x50)+25%x(1+1+1.99%x50)= 2.24

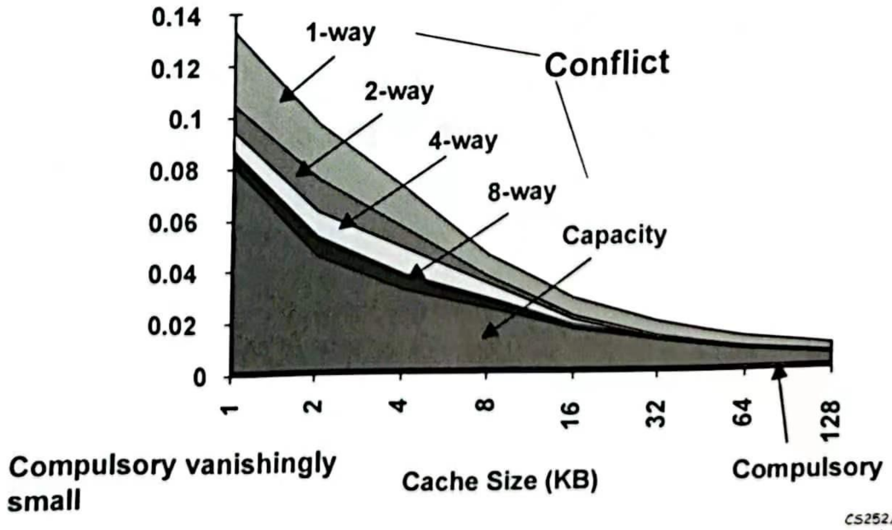
A.M.A.T = Thit + Imissrate · Tpenalty

Direct Mapping · Associative · Memory

Reducing Misses

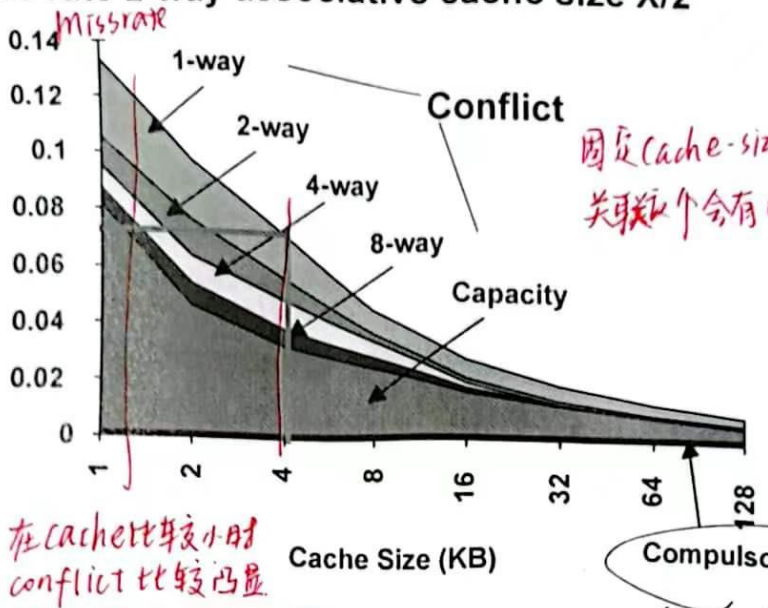
- Classifying Misses: 3 Cs
  - *Compulsory*—The first access to a block is not in the cache, so the block must be brought into the cache. Also called *cold start misses* or *first reference misses*.
    (Misses in even an Infinite Cache)
  - *Capacity*—If the cache cannot contain all the blocks needed during execution of a program, capacity misses will occur due to blocks being discarded and later retrieved.
    (Misses in Fully Associative Size X Cache)
  - *Conflict*—If block-placement strategy is set associative or direct mapped, conflict misses (in addition to compulsory & capacity misses) will occur because a block can be discarded and later retrieved if too many blocks map to its set. Also called *collision misses* or *interference misses*.
    (Misses in N-way Associative, Size X Cache)
- More recent, 4th "C":
  - *Coherence* - Misses caused by cache coherence.

3Cs Absolute Miss Rate (SPEC92)

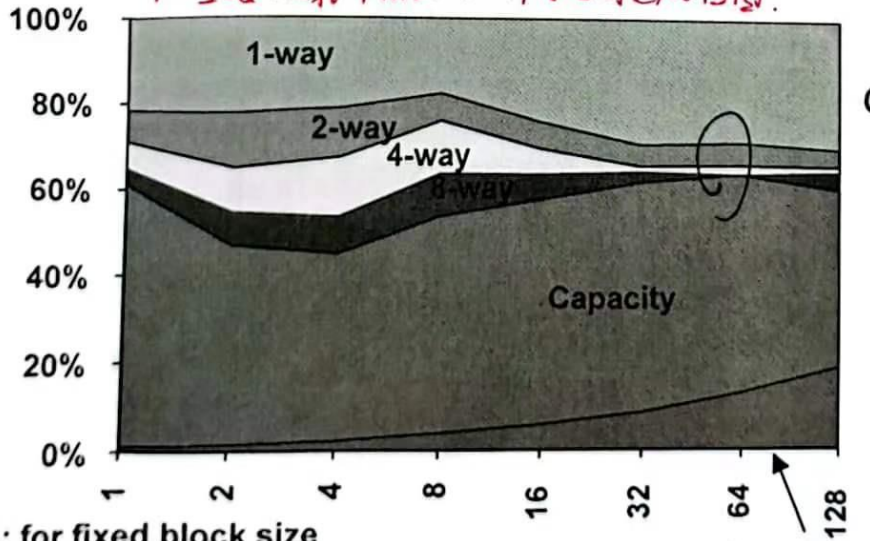


2:1 Cache Rule

miss rate 1-way associative cache size X
= miss rate 2-way associative cache size X/2



3Cs Relative Miss Rate



Flaws: for fixed block size
Good: insight => invention

How Can Reduce Misses?

- 3 Cs: Compulsory, Capacity, Conflict
- In all cases, assume total cache size not changed:
- What happens if:

1) Change Block Size:
   Which of 3Cs is obviously affected?

2) Change Associativity:
   Which of 3Cs is obviously affected?

3) Change Compiler:
   Which of 3Cs is obviously affected?

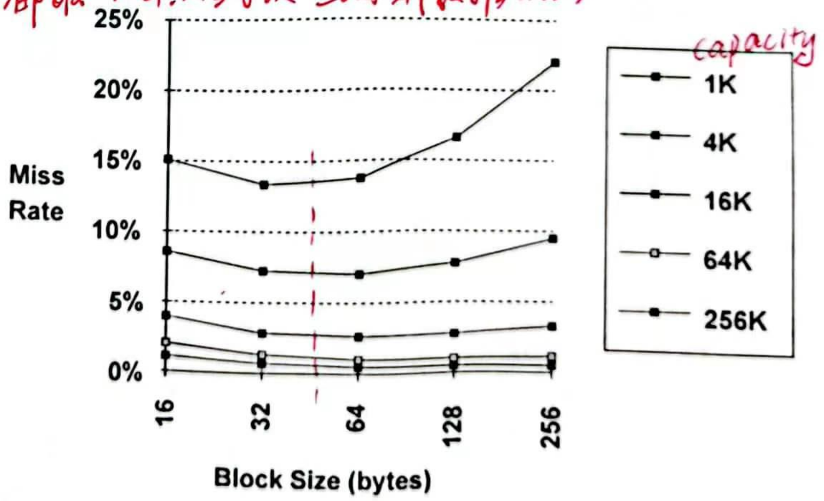
block 都有一个tag，标志着整个块的信息，一个block中的数据在memory中的位置是连续的
block 太小的时候受capacity的限制
block 略大可以发挥 locality 拿进来更多附近数据

1. Reduce Misses via Larger Block Size

但block过大，每miss一个数据
整块block都被换掉，又会导致一些捆绑数据 miss

←：破坏空间本地性

→：捆绑一起 一个miss 换掉整块



$\frac{1}{2^{tag}} \longrightarrow \frac{2}{2^{tag+1}}$ 个

2. Reduce Misses via Higher Associativity

- 2:1 Cache Rule:
  - Miss Rate DM cache size N - Miss Rate 2-way cache size N/2
- Beware: Execution time is only final measure!
  - Will Clock Cycle time increase?
  - Hill [1988] suggested hit time for 2-way vs. 1-way external cache +10%, internal + 2%

Miss Rate：并不是冷冰冰的概率比值
(由于数据使用的特点和分布) ⟺ 本地性

随着路数增加. missrate会↓

$A.M.A.T = T_{hit} + (1)_{missrate} \cdot T_{penalty}$

随着路数增加. 找到 hit 的时间↑

相互作用 先↑后↓

Example: Avg. Memory Access Time vs. Miss Rate

- Example: assume CCT = 1.10 for 2-way, 1.12 for 4-way, 1.14 for 8-way vs. CCT direct mapped

Cache Size (KB)	Associativity 1-way	2-way	4-way	8-way
1	2.33	2.15	2.07	2.01
2	1.98	1.86	1.76	1.68
4	1.72	1.67	1.61	1.53
8	1.46	1.48	1.47	1.43
16	1.29	1.32	1.32	1.32
32	1.20	1.24	1.25	1.27
64	1.14	1.20	1.21	1.23
128	1.10	1.17	1.18	1.20

单位 cycle

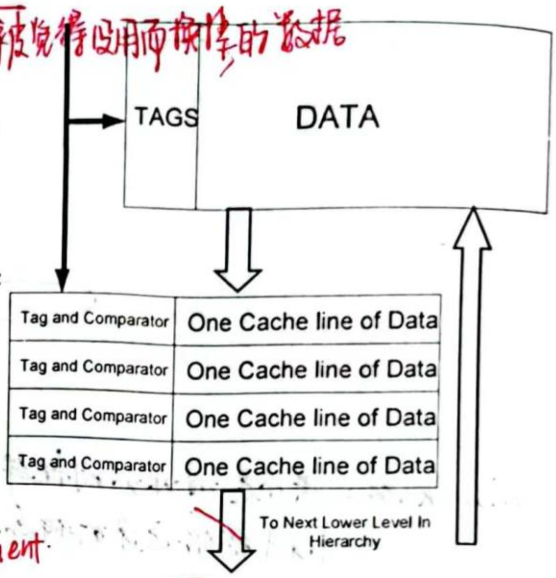
(Red means A.M.A.T. not improved by more associativity)

Miss
- compulsory 跟 block 大小有关
- conflict 跟关联度有关
- capacity 跟总容量有关

Write back & Write through.
如果某cache的数据 x —改→ x'
而memory 还未被修改的此cade
称为 dirty cache
dirty block (write through).

3. Reducing Misses via a "Victim Cache"

- How to combine fast hit time of direct mapped yet still avoid conflict misses?
- Add buffer to place data discarded from cache
- Jouppi [1990]: 4-entry victim cache removed 20% to 95% of conflicts for a 4 KB direct mapped data cache
- Used in Alpha, HP machines

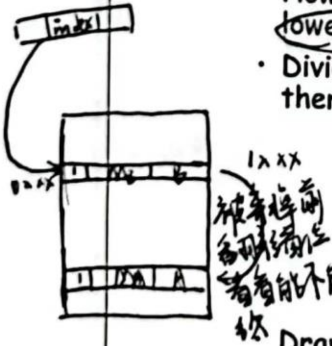


在工程实现中实现比较相似
Victim Cache vs. Write Buffer
IBM. HP. DEC → Digital Equipment.
victim cache是cache满了之后被替换掉的数据暂存到victim cache中如果下次再用可以拿回
Victim cache与memory 之间不是必须的，victim cache相当于被暂时存放cache废弃数据的垃圾桶，而write buffer要与memory相连
write buffer，用于write through.
在cache中修改了，肯定得同步修改到memory中，先放到buffer中找合适的机会写回.

4. Reducing Misses via "Pseudo-Associativity"

- How to combine fast hit time of Direct Mapped and have the lower conflict misses of 2-way SA cache?
- Divide cache: on a miss, check other half of cache to see if there, if so have a pseudo-hit (slow hit)

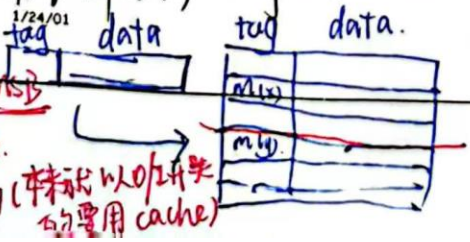
对全关联的一个改进 利用空位置



如果出现conflict 不先replace
在这之前翻转tag的首位用另一半空间

- Drawback: CPU pipeline is hard if hit takes 1 or 2 cycles
  - Better for caches not tied directly to processor (L2)
  - Used in MIPS R1000 L2 cache, similar in UltraSPARC

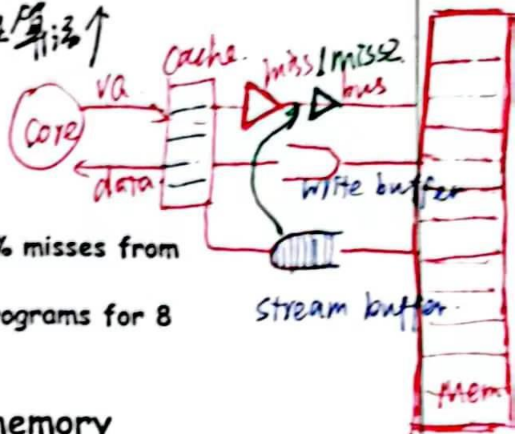
保持直接映射高速的同时
也利用另一半减少 Conflicts Miss

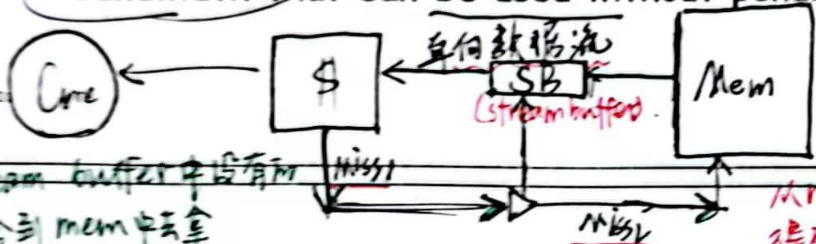


增加一个index MSB
防止冲突问题.
防止被占用的(本来就可以用的空闲的cache)

5. Reducing Misses by Hardware Prefetching of Instructions & Data

- E.g., Instruction Prefetching
  - Alpha 21064 fetches 2 blocks on a miss
  - Extra block placed in "stream buffer"
  - On miss check stream buffer
- Works with data blocks too:
  - Jouppi [1990] 1 data stream buffer got 25% misses from 4KB cache; 4 streams got 43%
  - Palacharla & Kessler [1994] for scientific programs for 8 streams got 50% to 70% of misses from 2 64KB, 4-way set associative caches
- Prefetching relies on having extra memory bandwidth that can be used without penalty





6. Reducing Misses by Software Prefetching Data

- Data Prefetch
  - Load data into register (HP PA-RISC loads)
  - Cache Prefetch: load into cache (MIPS IV, PowerPC, SPARC v. 9)
  - Special prefetching instructions cannot cause faults; a form of speculative execution
- Prefetching comes in two flavors:
  - Binding prefetch: Requests load directly into register.
    - Must be correct address and register!
  - Non-Binding prefetch: Load into cache.
    - Can be incorrect. Frees HW/SW to guess!
- Issuing Prefetch Instructions takes time
  - Is cost of prefetch issues < savings in reduced misses?
  - Higher superscalar reduces difficulty of issue bandwidth