Lecture 16: Reducing Cache Miss Penalty and Exploit Memory Parallelism

Critical work first, reads priority over writes, merging write buffer, non-blocking cache, stream buffer, and software prefetching

Improving Cache Performance

1. Reducing miss rates
   - Larger block size
   - larger cache size
   - higher associativity
   - victim caches
   - way prediction and Pseudoassociativity
   - compiler optimization
2. Reducing miss penalty
   - Multilevel caches
   - **critical word first**
   - **read miss first**
   - **merging write buffers**
3. Reducing miss penalty or miss rates via parallelism
   - Reduce miss penalty or miss rate by parallelism
   - Non-blocking caches
   - Hardware prefetching
   - Compiler prefetching
4. Reducing cache hit time
   - Small and simple caches
   - Avoiding address translation
   - Pipelined cache access
   - Trace caches

Early Restart and Critical Word First

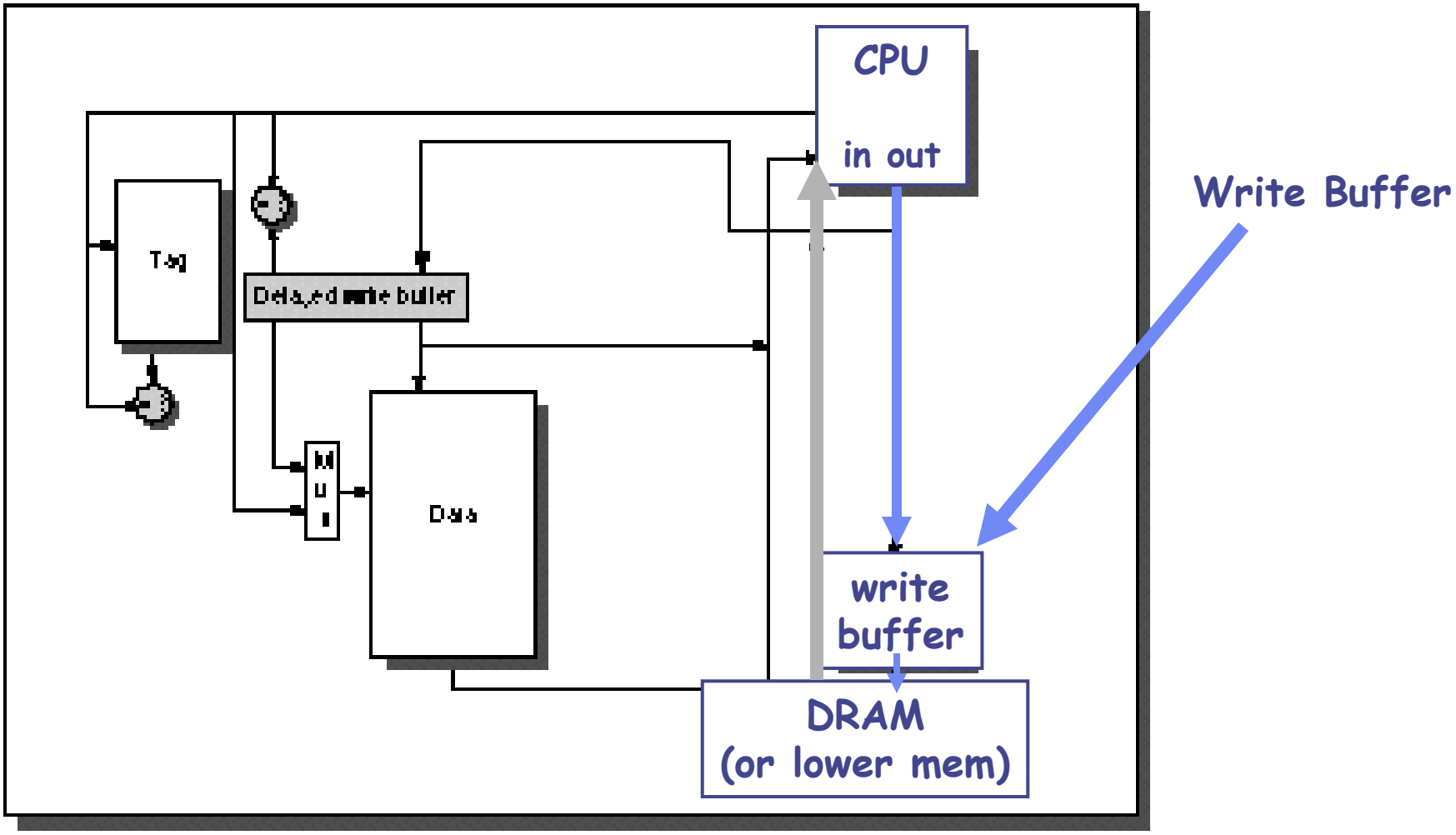
- Don't wait for full block to be loaded before restarting CPU
  - Early restart—As soon as the requested word of the block arrives, send it to the CPU and let the CPU continue execution
  - Critical Word First—Request the missed word first from memory and send it to the CPU as soon as it arrives; let the CPU continue execution while filling the rest of the words in the block. Also called wrapped fetch and requested word first
- Generally useful only in large blocks (relative to bandwidth)
- Good spatial locality may reduce the benefits of early restart, as the next sequential word may be needed anyway

block

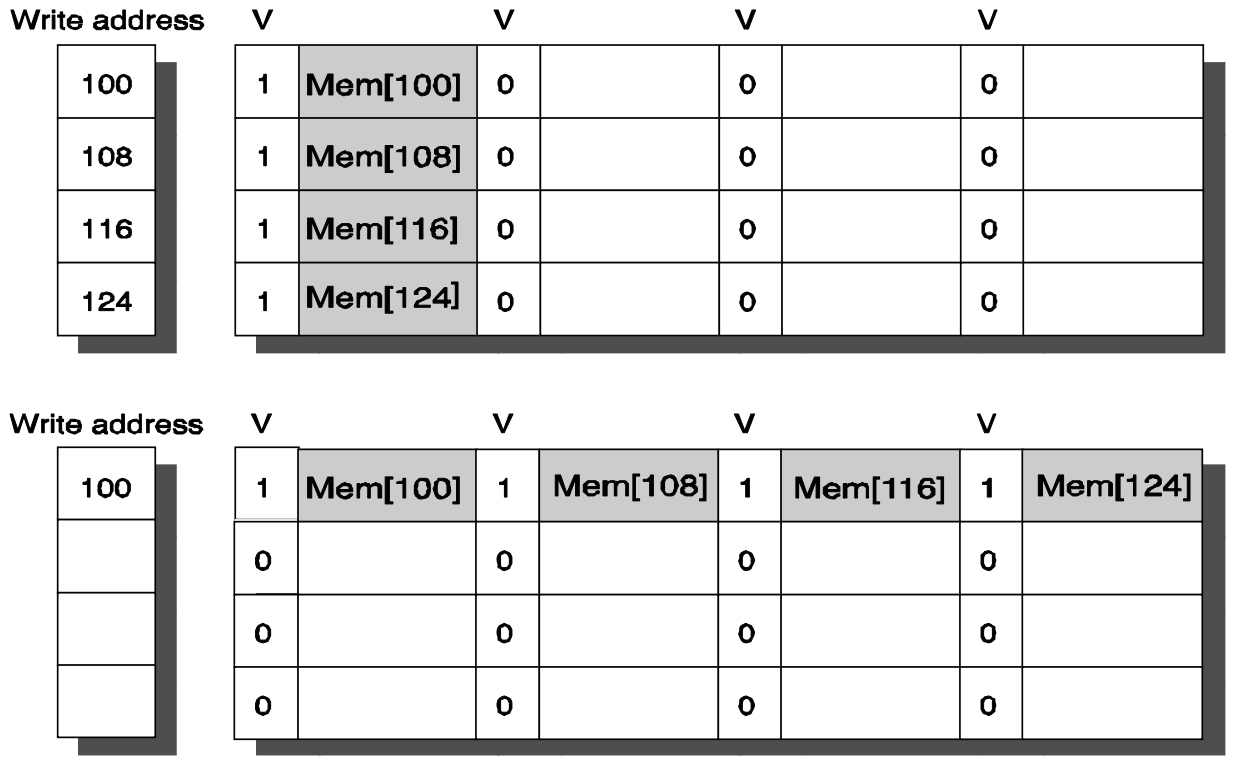
Read Priority over Write on Miss

- Write-through with write buffers offer RAW conflicts with main memory reads on cache misses
  - If simply wait for write buffer to empty, might increase read miss penalty (old MIPS 1000 by 50%)
  - Check write buffer contents before read; if no conflicts, let the memory access continue
  - Usually used with no-write allocate and a write buffer
- Write-back also want buffer to hold misplaced blocks
  - Read miss replacing dirty block
  - Normal: Write dirty block to memory, and then do the read
  - Instead copy the dirty block to a write buffer, then do the read, and then do the write
  - CPU stall less since restarts as soon as do read
  - Usually used with write allocate and a writeback buffer

Read Priority over Write on Miss



Merging Write Buffer



- Write merging: new written data into an existing block are merged
- Reduce stall for write (writeback) buffer being full
- Improve memory efficiency

Reducing Miss Penalty Summary

$$CPUtime = IC \times \left(CPI_{Execution} + \frac{Memory\ \ accesses}{Instruction} \times \textbf{Miss rate} \times Miss\ \ penalty \right) \times Clock\ \ cycle\ \ time$$

- Four techniques
  - Multi-level cache
  - Early Restart and Critical Word First on miss
  - Read priority over write
  - Merging write buffer
- Can be applied recursively to Multilevel Caches
  - Danger is that time to DRAM will grow with multiple levels in between
  - First attempts at L2 caches can make things worse, since increased worst case is worse

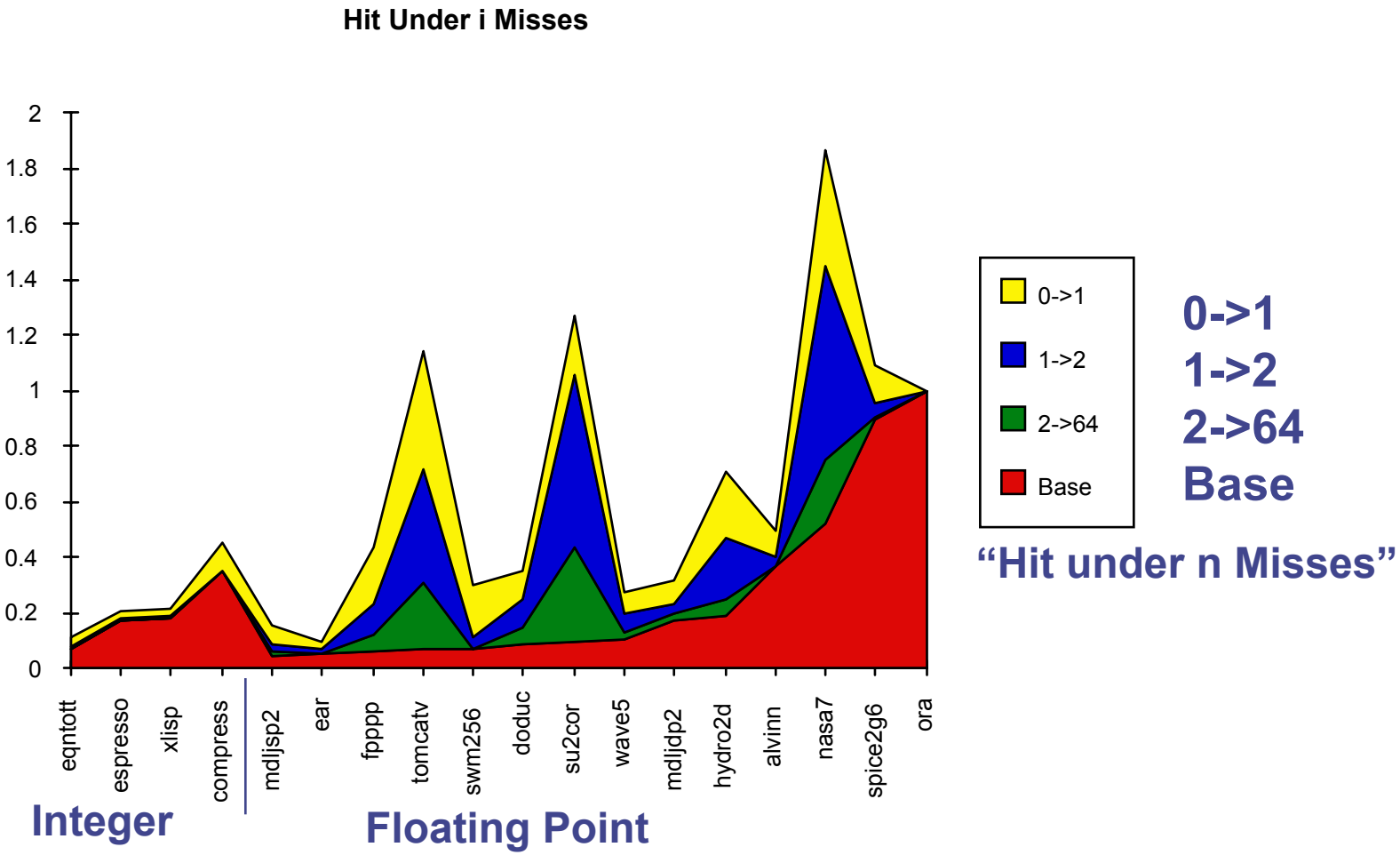
Improving Cache Performance

1. Reducing miss rates
   - Larger block size
   - larger cache size
   - higher associativity
   - victim caches
   - way prediction and Pseudoassociativity
   - compiler optimization
2. Reducing miss penalty
   - Multilevel caches
   - critical word first
   - read miss first
   - merging write buffers
3. Reducing miss penalty or miss rates via parallelism
   - **Reduce miss penalty or miss rate by parallelism**
   - **Non-blocking caches**
   - **Hardware prefetching**
   - **Compiler prefetching**
4. Reducing cache hit time
   - Small and simple caches
   - Avoiding address translation
   - Pipelined cache access
   - Trace caches

Non-blocking Caches to reduce stalls on misses

- Non-blocking cache or lockup-free cache allow data cache to continue to supply cache hits during a miss
  - Usually works with out-of-order execution
- "hit under miss" reduces the effective miss penalty by allowing one cache miss; processor keeps running until another miss happens
  - Sequential memory access is enough
  - Relative simple implementation
- "hit under multiple miss" or "miss under miss" may further lower the effective miss penalty by overlapping multiple misses
  - Implies memories support concurrency (parallel or pipelined)
  - Significantly increases the complexity of the cache controller
  - Requires muliple memory banks (otherwise cannot support)
  - Penium Pro allows 4 outstanding memory misses

Value of Hit Under Miss for SPEC

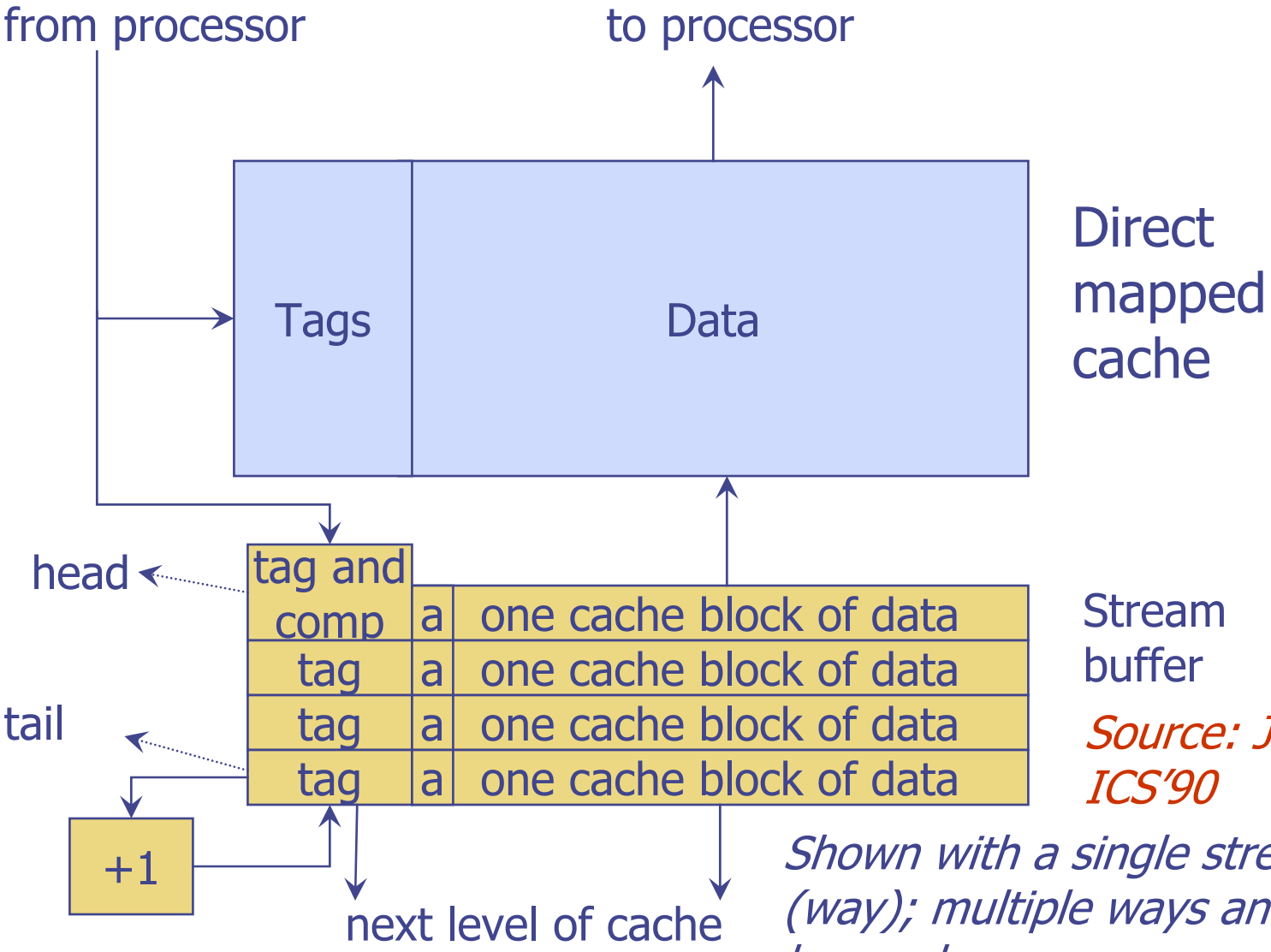


- FP programs on average: AMAT= 0.68 -> 0.52 -> 0.34 -> 0.26
- Int programs on average: AMAT= 0.24 -> 0.20 -> 0.19 -> 0.19
- 8 KB Data Cache, Direct Mapped, 32B block, 16 cycle miss

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

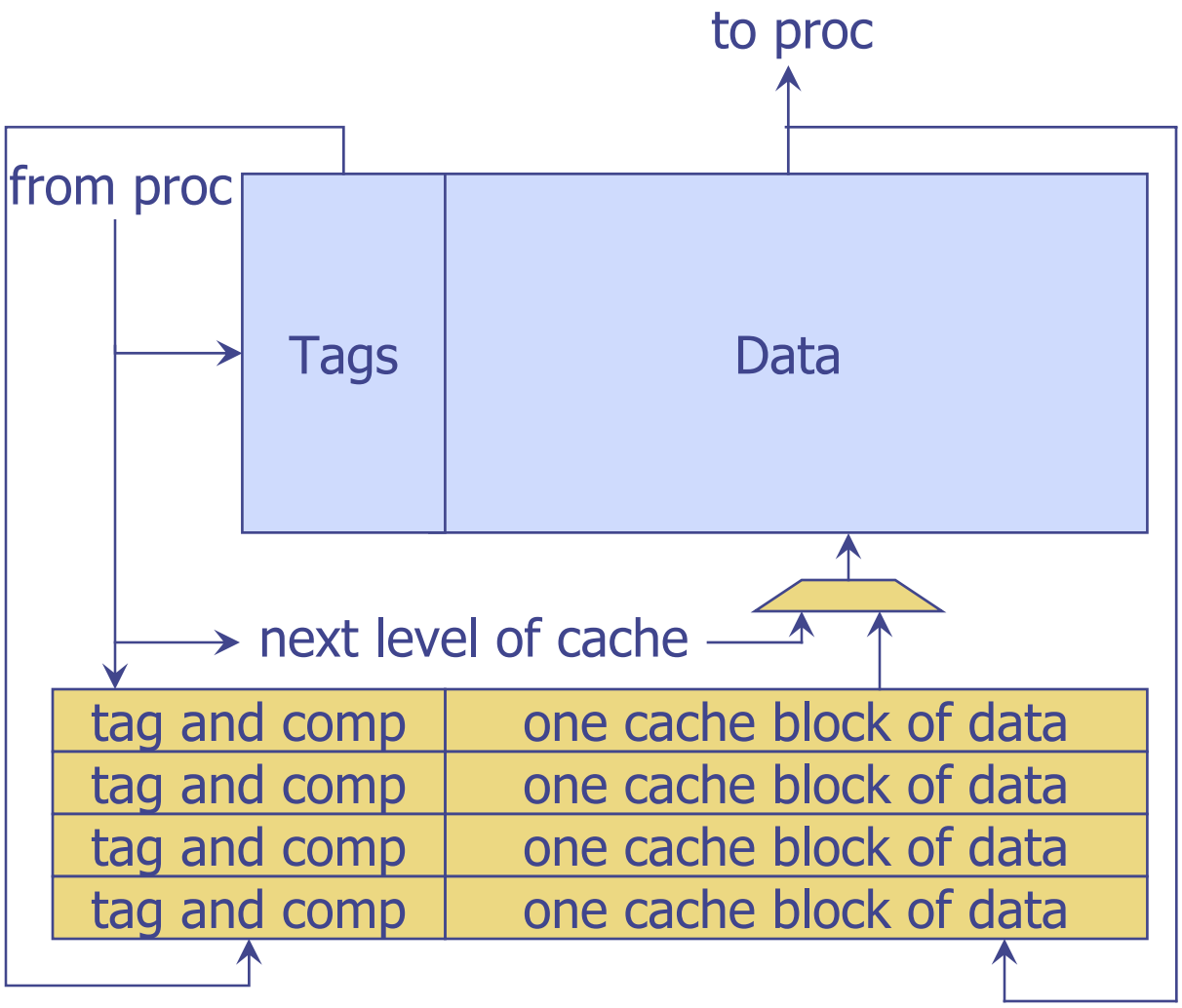
Stream Buffer Diagram



Source: Jouppi ICS'90

Shown with a single stream buffer (way); multiple ways and filter may be used

Victim Buffer Diagram



Direct mapped cache

Proposed in the same paper: *Jouppi ICS'90*

Victim cache, fully associative

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