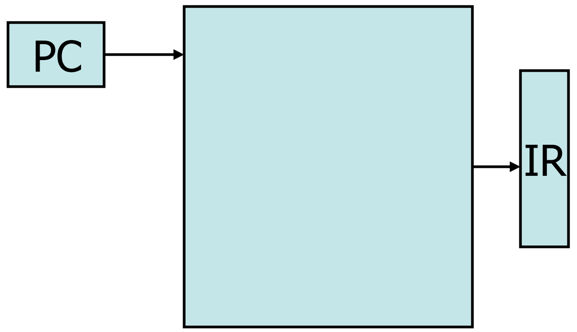
RISC Design: Memory System Design

Virendra Singh
Associate Professor
Computer Architecture and Dependable Systems Lab
Department of Electrical Engineering
Indian Institute of Technology Bombay
http://www.ee.iitb.ac.in/~viren/
E-mail: viren@ee.iitb.ac.in

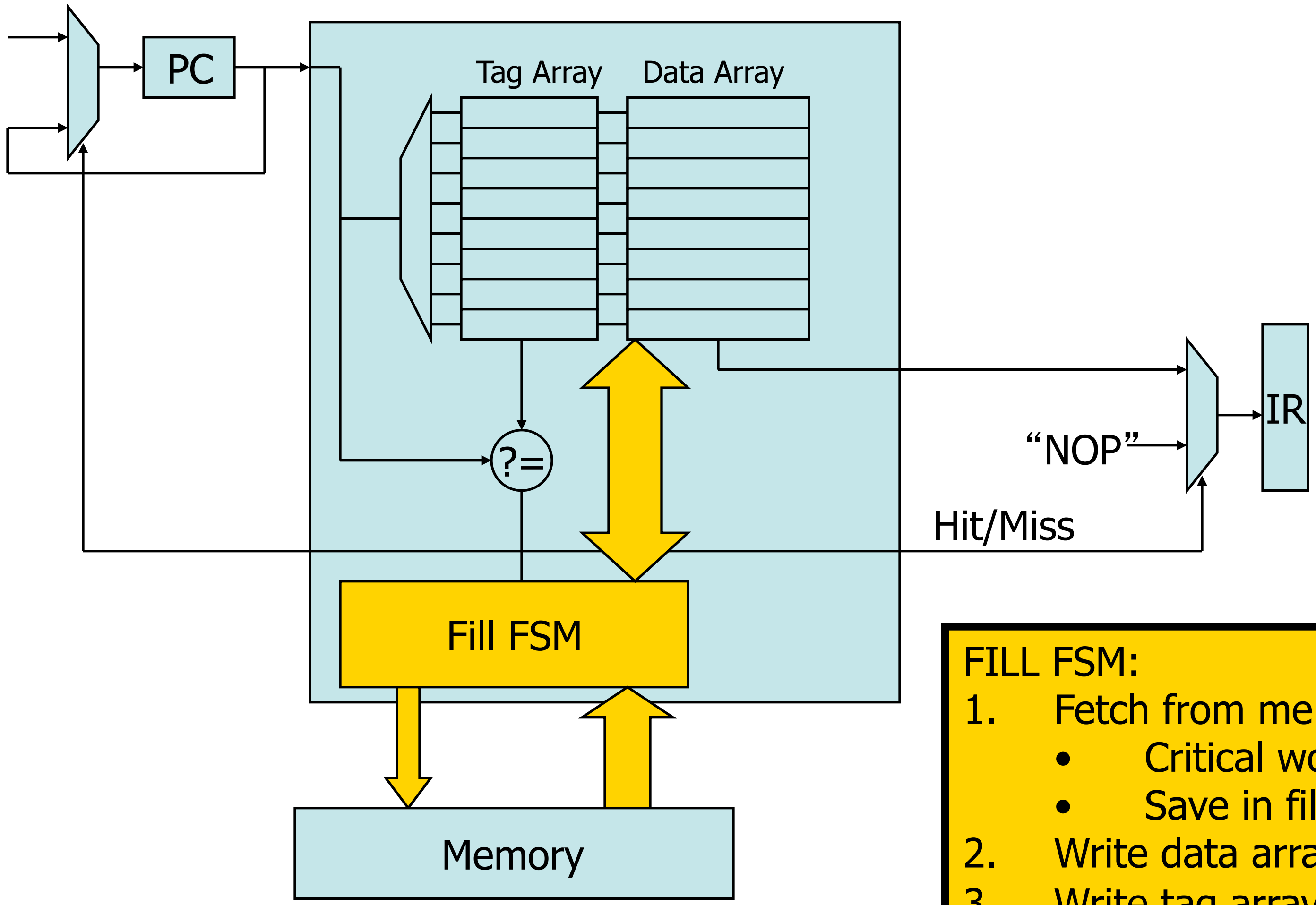
CP-226: Computer Architecture

Caches and Pipelining

- Instruction cache
  - No writes, so simpler
- Interface to pipeline:
  - Fetch address (from PC)
  - Supply instruction (to IR)
- What happens on a miss?
  - Stall pipeline; inject nop
  - Initiate cache fill from memory
  - Supply requested instruction, end stall condition



I-Caches and Pipelining



FILL FSM:

1. Fetch from memory
   - Critical word first
   - Save in fill buffer
2. Write data array
3. Write tag array
4. Miss condition ends

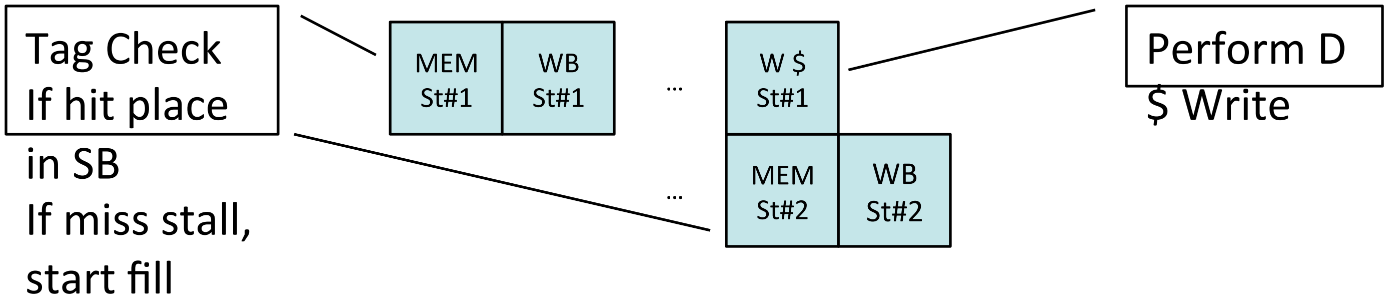
D-Caches and Pipelining

- Pipelining loads from cache
  - Hit/Miss signal from cache
  - Stalls pipeline or inject NOPs?
    - Hard to do in current real designs, since wires are too slow for global stall signals
  - Instead, treat more like branch misprediction
    - Cancel/flush pipeline
    - Restart when cache fill logic is done

D-Caches and Pipelining

- Stores more difficult
  - MEM stage:
    - Perform tag check
    - Only enable write on a hit
    - On a miss, must not write (data corruption)
  - Problem:
    - Must do tag check and data array access sequentially
    - This will hurt cycle time or force extra pipeline stage
    - Extra pipeline stage delays loads as well: IPC hit!

Solution: Pipelining Writes



- Store #1 performs tag check only in MEM stage
  - <value, address, cache way> placed in store buffer (SB)
- When store #2 reaches MEM stage
  - Store #1 writes to data cache
- In the meantime, must handle RAW to store buffer
  - Pending write in SB to address A
  - Newer loads must check SB for conflict
  - Stall/flush SB, or forward directly from SB
- Any load miss must also flush SB first
  - Otherwise SB D$ write may be to wrong line
- Can expand to >1 entry to overlap store misses

Summary

- Memory technology
- Memory hierarchy
  - Temporal and spatial locality
- Caches
  - Placement
  - Identification
  - Replacement
  - Write Policy
- Pipeline integration of caches

Performance

CPU execution time = (CPU clock cycles + memory stall cycles) x Clock Cycle time

Memory Stall cycles = Number of misses x miss penalty

= IC x misses/Instruction x miss penalty

=IC x memory access/instruction x miss rate x miss penalty

Thank You