Main Memory Supporting Caches

Use DRAMs for main memory
- Fixed width (e.g., 1 word)
- Connected by fixed-width clocked bus
  - Bus clock is typically slower than CPU clock

Example cache block read
- 1 bus cycle for address transfer
- 15 bus cycles per DRAM access
- 1 bus cycle per data transfer

For 4-word block, 1-word-wide DRAM
- Miss penalty = 1 + 4×15 + 4×1 = 65 bus cycles
- Bandwidth = 16 bytes / 65 cycles = 0.25 B/cycle
Increasing Memory Bandwidth

4-word wide memory
- Miss penalty = 1 + 15 + 1 = 17 bus cycles
- Bandwidth = 16 bytes / 17 cycles = 0.94 B/cycle

4-bank interleaved memory
- Miss penalty = 1 + 15 + 4×1 = 20 bus cycles
- Bandwidth = 16 bytes / 20 cycles = 0.8 B/cycle
Advanced DRAM Organization

Bits in a DRAM are organized as a rectangular array
  - DRAM accesses an entire row
  - Burst mode: supply successive words from a row with reduced latency

Double data rate (DDR) DRAM
  - Transfer on both rising and falling clock edges

Quad data rate (QDR) DRAM
  - Separate DDR inputs and outputs
Measuring Cache Performance

Components of CPU time
- Program execution cycles
  - Includes cache hit time
- Memory stall cycles
  - Mainly from cache misses

With simplifying assumptions:

\[
\text{Memory stall cycles} = \frac{\text{Memory accesses}}{\text{Program}} \times \text{Miss rate} \times \text{Miss penalty}
\]

\[
= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Misses}}{\text{Instruction}} \times \text{Miss penalty}
\]
Cache Performance Example

Given
- I-cache miss rate = 2%
- D-cache miss rate = 4%
- Miss penalty = 100 cycles
- Base CPI (ideal cache) = 2
- Load & stores are 36% of instructions

Miss cycles per instruction
- I-cache: $0.02 \times 100 = 2$
- D-cache: $0.36 \times 0.04 \times 100 = 1.44$

Actual CPI = $2 + 2 + 1.44 = 5.44$
- Ideal CPU is $5.44/2 = 2.72$ times faster
Average Access Time

Hit time is also important for performance

Average memory access time (AMAT)

- AMAT = Hit time + Miss rate × Miss penalty

Example

- CPU with 1ns clock, hit time = 1 cycle, miss penalty = 20 cycles, I-cache miss rate = 5%
  - AMAT = 1 + 0.05 × 20 = 2ns
    - 2 cycles per instruction
Performance Summary

When CPU performance increased
  - Miss penalty becomes more significant

Decreasing base CPI
  - Greater proportion of time spent on memory stalls

Increasing clock rate
  - Memory stalls account for more CPU cycles

Can’t neglect cache behavior when evaluating system performance
Associative Caches

Fully associative
- Allow a given block to go in any cache entry
- Requires all entries to be searched at once
- Comparator per entry (expensive)

$n$-way set associative
- Each set contains $n$ entries
- Block number determines which set
  - $(\text{Block number}) \mod (\#\text{Sets in cache})$
- Search all entries in a given set at once
- $n$ comparators (less expensive)
Associative Cache Example

Direct mapped

Set associative

Fully associative

<table>
<thead>
<tr>
<th>Block #</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Set #</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Tag     | 1 | 2 |
| Search  |   |   |   |   |   |   |   |   |
For a cache with 8 entries:

### One-way set associative (direct mapped)

- Block 0: Tag, Data
- Block 1: Tag, Data
- Block 2: Tag, Data
- Block 3: Tag, Data
- Block 4: Tag, Data
- Block 5: Tag, Data
- Block 6: Tag, Data
- Block 7: Tag, Data

### Two-way set associative

- Set 0: Tag, Data, Tag, Data
- Set 1: Tag, Data, Tag, Data
- Set 2: Tag, Data, Tag, Data
- Set 3: Tag, Data, Tag, Data

### Four-way set associative

- Set 0: Tag, Data, Tag, Data, Tag, Data, Tag, Data
- Set 1: Tag, Data, Tag, Data, Tag, Data, Tag, Data

### Eight-way set associative (fully associative)

- Tag, Data, Tag, Data, Tag, Data, Tag, Data, Tag, Data, Tag, Data, Tag, Data, Tag, Data, Tag, Data, Tag, Data
Associativity Example

Compare 4-block caches
- Direct mapped, 2-way set associative, fully associative
- Block access sequence: 0, 8, 0, 6, 8

Direct mapped

<table>
<thead>
<tr>
<th>Block address</th>
<th>Cache index</th>
<th>Hit/miss</th>
<th>Cache content after access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>miss</td>
<td>Mem[0]</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>miss</td>
<td>Mem[8]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>miss</td>
<td>Mem[0]</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>miss</td>
<td>Mem[0]</td>
</tr>
</tbody>
</table>
## Associativity Example

### 2-way set associative

<table>
<thead>
<tr>
<th>Block address</th>
<th>Cache index</th>
<th>Hit/miss</th>
<th>Cache content after access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Set 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>miss</td>
<td>Mem[0]</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>miss</td>
<td>Mem[0]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>hit</td>
<td>Mem[0]</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>miss</td>
<td>Mem[0]</td>
</tr>
</tbody>
</table>

### Fully associative

<table>
<thead>
<tr>
<th>Block address</th>
<th>Hit/miss</th>
<th>Cache content after access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Set 0</td>
</tr>
<tr>
<td>0</td>
<td>miss</td>
<td>Mem[0]</td>
</tr>
<tr>
<td>8</td>
<td>miss</td>
<td>Mem[0]</td>
</tr>
<tr>
<td>0</td>
<td>hit</td>
<td>Mem[0]</td>
</tr>
</tbody>
</table>
How Much Associativity

Increased associativity decreases miss rate
  - But with diminishing returns

Simulation of a system with 64KB D-cache, 16-word blocks, SPEC2000
  - 1-way: 10.3%
  - 2-way: 8.6%
  - 4-way: 8.3%
  - 8-way: 8.1%
Set Associative Cache Organization
Replacement Policy

Direct mapped: no choice

Set associative
- Prefer non-valid entry, if there is one
- Otherwise, choose among entries in the set

Least-recently used (LRU)
- Choose the one unused for the longest time
  - Simple for 2-way, manageable for 4-way, too hard beyond that

Random
- Gives approximately the same performance as LRU for high associativity
Multilevel Caches

Primary cache attached to CPU
  - Small, but fast

Level-2 cache services misses from primary cache
  - Larger, slower, but still faster than main memory

Main memory services L-2 cache misses

Some high-end systems include L-3 cache
Multilevel Cache Example

Given
- CPU base CPI = 1, clock rate = 4GHz
- Miss rate/instruction = 2%
- Main memory access time = 100ns

With just primary cache
- Miss penalty = 100ns/0.25ns = 400 cycles
- Effective CPI = $1 + 0.02 \times 400 = 9$
Now add L-2 cache
   - Access time = 5ns
   - Global miss rate to main memory = 0.5%

Primary miss with L-2 hit
   - Penalty = 5ns/0.25ns = 20 cycles

Primary miss with L-2 miss
   - Extra penalty = 500 cycles

\[\text{CPI} = 1 + 0.02 \times 20 + 0.005 \times 400 = 3.4\]

Performance ratio = 9/3.4 = 2.6
Multilevel Cache Considerations

Primary cache
- Focus on minimal hit time

L-2 cache
- Focus on low miss rate to avoid main memory access
- Hit time has less overall impact

Results
- L-1 cache usually smaller than a single cache
- L-1 block size smaller than L-2 block size