Caching and Virtual Memory
Last Time

• Flexible Address Translation
  – Segmentation + paged translation
  – Multi-level paged translation
  – Hashing

• Efficient Address Translation
  – Translation Lookaside Buffers (TLBs)
  – Virtually addressed and physically addressed caches
Main Points

• Cache concept
  – Hardware vs. software caches

• When caches work and when they don’t
  – Spatial/temporal locality vs. Zipf workloads

• Cache replacement policies
Multicore and Hyperthreading

• Modern CPU has several functional units
  – Instruction decode
  – Arithmetic/branch
  – Floating point
  – Instruction/data cache
  – TLB
• Multicore: replicate functional units (i7: 4)
  – Share second/third level cache, second level TLB
• Hyperthreading: logical processors that share functional units (i7: 2)
  – Better functional unit utilization during memory stalls
• No difference from the OS/programmer perspective
  – Except for performance, affinity, ...
Definitions

• Cache
  – Copy of data that is faster to access than the original
  – Hit: if cache has copy
  – Miss: if cache does not have copy

• Cache block
  – Unit of cache storage (multiple memory locations)

• Temporal locality
  – Programs tend to reference the same memory locations multiple times
  – Example: instructions in a loop

• Spatial locality
  – Programs tend to reference nearby locations
  – Example: data in a loop
Cache Concept (Read)

fetch address

Cache

address in cache?

yes

no

fetch address

value stored at address
Cache Concept (Write)

- Store value at address
- Store value at address
- Address in cache?
  - Yes
  - Store value in cache
  - If write through
  - Store value at address
- No
- Fetch address

Write through: changes sent immediately to next level of storage.

Write back: changes stored in cache until cache block is replaced.
## Memory Hierarchy

<table>
<thead>
<tr>
<th>Cache</th>
<th>Hit Cost</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st level cache/first level TLB</td>
<td>1 ns</td>
<td>64 KB</td>
</tr>
<tr>
<td>2nd level cache/second level TLB</td>
<td>4 ns</td>
<td>256 KB</td>
</tr>
<tr>
<td>3rd level cache</td>
<td>12 ns</td>
<td>2 MB</td>
</tr>
<tr>
<td>Memory (DRAM)</td>
<td>100 ns</td>
<td>10 GB</td>
</tr>
<tr>
<td>Data center memory (DRAM)</td>
<td>100 μs</td>
<td>100 TB</td>
</tr>
<tr>
<td>Local non-volatile memory</td>
<td>100 μs</td>
<td>100 GB</td>
</tr>
<tr>
<td>Local disk</td>
<td>10 ms</td>
<td>1 TB</td>
</tr>
<tr>
<td>Data center disk</td>
<td>10 ms</td>
<td>100 PB</td>
</tr>
<tr>
<td>Remote data center disk</td>
<td>200 ms</td>
<td>1 XB</td>
</tr>
</tbody>
</table>

i7 has 8MB as shared 3\textsuperscript{rd} level cache; 2\textsuperscript{nd} level cache is per-core
Hardware Design Principle

The bigger the memory, the slower the memory
Address Translation with TLB

- Processor
- Virtual Address
- Translation Box
  - virtual page in TLB? yes → Physical Address
  - no → valid page table entry?
    - yes → Physical Memory
    - no → raise exception
- Instruction fetch or data read/write (untranslated)
Virtually Addressed Caches

![Diagram showing the process of memory translation and cache usage.]

- Instruction fetch or data read/write (untranslated)
- Virtual Address
- Processor
- Translation Box
  - virtual page in TLB? yes
  - location in virtual cache?
  - valid page table entry?
  - Physical Address
    - yes
    - no
      - raise exception
- Physical Memory
Questions

• With a virtual cache, what do we need to do on a context switch?

• What if the virtual cache > page size?
  – Page size: 4KB (x86)
  – First level cache size: 64KB (i7)
  – Cache block size: 32 bytes
Aliasing

• Alias: two (or more) virtual cache entries that refer to the same physical memory
  – What if we modify one alias and then context switch?

• Typical solution
  – On a write, lookup virtual cache and TLB in parallel
  – Physical address from TLB used to check for aliases
Memory Hierarchy

<table>
<thead>
<tr>
<th>Cache</th>
<th>Hit Cost</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st level cache/first level TLB</td>
<td>1 ns</td>
<td>64 KB</td>
</tr>
<tr>
<td>2nd level cache/second level TLB</td>
<td>4 ns</td>
<td>256 KB</td>
</tr>
<tr>
<td>3rd level cache</td>
<td>12 ns</td>
<td>2 MB</td>
</tr>
<tr>
<td>Memory (DRAM)</td>
<td>100 ns</td>
<td>10 GB</td>
</tr>
<tr>
<td>Data center memory (DRAM)</td>
<td>100 µs</td>
<td>100 TB</td>
</tr>
<tr>
<td>Local non-volatile memory</td>
<td>100 µs</td>
<td>100 GB</td>
</tr>
<tr>
<td>Local disk</td>
<td>10 ms</td>
<td>1 TB</td>
</tr>
<tr>
<td>Data center disk</td>
<td>10 ms</td>
<td>100 PB</td>
</tr>
<tr>
<td>Remote data center disk</td>
<td>200 ms</td>
<td>1 XB</td>
</tr>
</tbody>
</table>

i7 has 8MB as shared 3\textsuperscript{rd} level cache; 2\textsuperscript{nd} level cache is per-core
Translation on a Modern Processor

Translation Box

Virtual Address

Processor

Location in virtual cache?

no

虚 pages in TLB?

yes

Physical Address

Valid page table entry?

no

raise exception

Location in physical cache?

Physical Memory

Location in physical cache?

no

no

Instruction fetch or data read/write (untranslated)
Question

• What is the cost of a first level TLB miss?
  – Second level TLB lookup

• What is the cost of a second level TLB miss?
  – x86: 2-4 level page table walk

• How expensive is a 4-level page table walk on a modern processor?
Working Set Model

- Working Set: set of memory locations that need to be cached for reasonable cache hit rate
- Thrashing: when system has too small a cache
Phase Change Behavior

- Programs can change their working set
- Context switches also change working set

![Graph showing cache hit rate over time with points labeled Phase change and new equilibrium]
Zipf Distribution

• Caching behavior of many systems are not well characterized by the working set model
• An alternative is the Zipf distribution
  – Popularity $\sim 1/k^c$, for kth most popular item, $1 < c < 2$
Zipf Distribution

- X-axis: Rank
- Y-axis: Popularity
Zipf Examples

• Web pages
• Movies
• Library books
• Words in text
• Salaries
• City population
• ...

Common thread: popularity is self-reinforcing
Zipf and Caching

The graph shows the relationship between cache hit rate and cache size (on a log scale). The hit rate increases linearly as the cache size grows from .001% to all.

- Cache Hit Rate
- Cache Size (log scale)
Cache Lookup: Fully Associative

match at any address?

address

<table>
<thead>
<tr>
<th>address</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>=?</td>
<td></td>
</tr>
<tr>
<td>=?</td>
<td></td>
</tr>
<tr>
<td>=?</td>
<td></td>
</tr>
<tr>
<td>=?</td>
<td></td>
</tr>
<tr>
<td>=?</td>
<td></td>
</tr>
</tbody>
</table>

yes

return value
Cache Lookup: Direct Mapped

\[
\begin{array}{|c|c|}
\hline
\text{address} & \text{value} \\
\hline
\text{hash(address)} & \text{=}?
\text{match at hash(address)?}
\hline
\end{array}
\]

\[
\text{yes} \\
\text{return value}
\]
Cache Lookup: Set Associative

1. Compute hash(address)
2. Check if match at hash(address)?
3. If yes, return value

<table>
<thead>
<tr>
<th>address</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0053</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>0x120d</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Page Coloring

• What happens when cache size >> page size?
  – Direct mapped or set associative
  – Multiple pages map to the same cache line

• OS page assignment matters!
  – Example: 8MB cache, 4KB pages
  – 1 of every 2K pages lands in same place in cache

• What should the OS do?
Cache Replacement Policy

• On a cache miss, how do we choose which entry to replace?
  – Assuming the new entry is more likely to be used in the near future
  – In direct mapped caches, not an issue!

• Policy goal: reduce cache misses
  – Improve expected case performance
  – Also: reduce likelihood of very poor performance
A Simple Policy

• Random?
  – Replace a random entry

• FIFO?
  – Replace the entry that has been in the cache the longest time
  – What could go wrong?
Worst case for FIFO is if program strides through memory that is larger than the cache
MIN, LRU, LFU

• MIN
  – Replace the cache entry that will not be used for the longest time into the future
  – Optimality proof based on exchange: if evict an entry used sooner, that will trigger an earlier cache miss

• Least Recently Used (LRU)
  – Replace the cache entry that has not been used for the longest time in the past
  – Approximation of MIN

• Least Frequently Used (LFU)
  – Replace the cache entry used the least often (in the recent past)
LRU/MIN for Sequential Scan

<table>
<thead>
<tr>
<th>Reference</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td></td>
<td>E</td>
<td></td>
<td></td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td></td>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td></td>
<td>A</td>
<td></td>
<td>E</td>
<td></td>
<td></td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td></td>
<td>B</td>
<td>A</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>D</td>
<td></td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reference</th>
<th>A</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td>+</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td></td>
<td></td>
<td>+</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>D</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>C</td>
<td>B</td>
<td>D</td>
<td>A</td>
<td>D</td>
<td>E</td>
<td>D</td>
<td>A</td>
<td>E</td>
<td>B</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>-----------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>B</td>
<td></td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>C</td>
<td></td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>D</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FIFO**

<table>
<thead>
<tr>
<th>Reference</th>
<th>A</th>
<th>B</th>
<th>A</th>
<th>C</th>
<th>B</th>
<th>D</th>
<th>A</th>
<th>D</th>
<th>E</th>
<th>D</th>
<th>A</th>
<th>E</th>
<th>B</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**MIN**

<table>
<thead>
<tr>
<th>Reference</th>
<th>A</th>
<th>B</th>
<th>A</th>
<th>C</th>
<th>B</th>
<th>D</th>
<th>A</th>
<th>D</th>
<th>E</th>
<th>D</th>
<th>A</th>
<th>E</th>
<th>B</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Belady’s Anomaly

<table>
<thead>
<tr>
<th>Reference</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>D</td>
<td>E</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>A</td>
<td></td>
<td>+</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>B</td>
<td></td>
<td></td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FIFO (4 slots)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>