Cache Organization

Daniel Sanchez
Computer Science and Artificial Intelligence Laboratory
M.I.T.

http://www.csg.csail.mit.edu/6.823
Snow day makeup lecture this Friday 2/24
  - 1-2:30p 37-212

We will discuss material at end of L02 slides (simple Princeton pipeline) in L06
Performance of high-speed computers is usually limited by memory bandwidth & latency

- **Latency (time for a single access)**
  Memory access time $>>$ Processor cycle time

- **Bandwidth (number of accesses per unit time)**
  if fraction $m$ of instructions access memory,
  $\Rightarrow 1+m$ memory references / instruction
  $\Rightarrow \text{CPI} = 1$ requires $1+m$ memory ref/s / cycle
Memory Technology

• Early machines used a variety of memory technologies
  – Manchester Mark I used CRT Memory Storage
  – EDVAC used a mercury delay line

• Core memory was first large scale reliable main memory
  – Invented by Forrester in late 40s at MIT for Whirlwind project
  – Bits stored as magnetization polarity on small ferrite cores threaded onto 2 dimensional grid of wires

• First commercial DRAM was Intel 1103
  – 1Kbit of storage on single chip
  – charge on a capacitor used to hold value

• Semiconductor memory quickly replaced core in 1970s
  – Intel formed to exploit market for semiconductor memory

• Flash memory
  – Slower, but denser than DRAM. Also non-volatile, but with wearout issues

• Phase change memory (PCM) a possibility for the future
  – Slightly slower, but much denser than DRAM and non-volatile
DRAM Architecture

- Bits stored in 2-dimensional arrays on chip
- Modern chips have around 8 logical banks on each chip
  - each logical bank physically implemented as many smaller arrays
DRAM timing

DRAM Spec:
CL, tRCD, tRP, tRAS, e.g., 9-9-9-24
Processor-DRAM Gap (latency)

Four-issue 2GHz superscalar accessing 100ns DRAM could execute 800 instructions during time for one memory access!
Little’s Law

Throughput \((T) = \frac{\text{Number in Flight} \,(N)}{\text{Latency} \,(L)}\)

Example:

--- Assume infinite-bandwidth memory
--- 100 cycles / memory reference
--- 1 + 0.2 memory references / instruction

\[ \Rightarrow \text{Table size} = 1.2 \times 100 = 120 \text{ entries} \]

120 independent memory operations in flight!
Basic Static RAM Cell

6-Transistor SRAM Cell

- Write:
  1. Drive bit lines (bit=1, \overline{bit}=0)
  2. Select word line
- Read:
  1. Precharge bit and \overline{bit} to Vdd
  2. Select word line
  3. Cell pulls one bit line low
  4. Column sense amp detects difference between bit & \overline{bit}
Multilevel Memory

Strategy: **Reduce** average latency using small, fast memories called caches.

Caches are a mechanism to reduce memory latency based on the empirical observation that the patterns of memory references made by a processor are often highly predictable:

<table>
<thead>
<tr>
<th>PC</th>
<th>Instruction</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>96</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td><code>ADD r2, r1, r1</code></td>
<td></td>
</tr>
<tr>
<td>104</td>
<td><code>SUBI r3, r3, #1</code></td>
<td></td>
</tr>
<tr>
<td>108</td>
<td><code>BNEZ r3, loop</code></td>
<td></td>
</tr>
<tr>
<td>112</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Typical Memory Reference Patterns

- Instruction fetches
- Stack accesses
- Data accesses

- n loop iterations
- subroutine call
- subroutine return
- argument access
- vector access
- scalar accesses
Common Predictable Patterns

Two predictable properties of memory references:

- **Temporal Locality**: If a location is referenced it is likely to be referenced again in the near future.

- **Spatial Locality**: If a location is referenced it is likely that locations near it will be referenced in the near future.
Memory Hierarchy

On a data access:

hit (data ∈ fast memory) ⇒ low latency access
miss (data ∉ fast memory) ⇒ long latency access (DRAM)
Management of Memory Hierarchy

- **Small/fast storage, e.g., registers**
  - Address usually specified in instruction
  - Generally implemented directly as a register file
    - but hardware might do things behind software’s back, e.g., stack management, register renaming

- **Large/slower storage, e.g., memory**
  - Address usually computed from values in register
  - Generally implemented as a cache hierarchy
    - hardware decides what is kept in fast memory
    - but software may provide “hints”, e.g., don’t cache or prefetch
Inside a Cache

How many bits are needed in tag?

Enough to uniquely identify block
Cache Algorithm (Read)

Look at Processor Address, search cache tags to find match. Then either

- Found in cache a.k.a. HIT
  - Return copy of data from cache

- Not in cache a.k.a. MISS
  - Read block of data from Main Memory
  - Wait ...
  - Return data to processor and update cache

Q: Which line do we replace?
Direct-Mapped Cache

What is a bad reference pattern?

Strided at size of cache
Direct Map Address Selection
higher-order vs. lower-order address bits

Why might this be undesirable?

Spatially local blocks conflict
Hashed Address Selection

What are the tradeoffs of hashing?

**Good:** Regular strides don’t conflict
**Bad:** Hash adds latency
Tag is larger
2-Way Set-Associative Cache
Fully Associative Cache

Diagram illustrating the structure of a fully associative cache, showing the flow of data and the comparison of tags and data blocks to determine a hit.
Placement Policy

Block Number

Memory

Set Number

Cache

Direct Mapped

(2-way) Set Associative

Fully Associative

block 12 can be placed only into block 4

(12 mod 8)

anywhere in set 0

(12 mod 4)

Anywhere
Improving Cache Performance

Average memory access time = 
Hit time + Miss rate x Miss penalty

To improve performance:
- reduce the hit time
- reduce the miss rate (e.g., larger, better policy)
- reduce the miss penalty (e.g., L2 cache)

What is the simplest design strategy?

Biggest cache that doesn’t increase hit time past 1-2 cycles
(approx 8-32KB in modern technology)
[design issues more complex with out-of-order superscalar processors]
Causes for Cache Misses

• **Compulsory:**
  first-reference to a block *a.k.a.* cold start misses
  - misses that would occur even with infinite cache

• **Capacity:**
  cache is too small to hold all data the program needs
  - misses that would occur even under perfect placement & replacement policy

• **Conflict:**
  misses from collisions due to block-placement strategy
  - misses that would not occur with full associativity
# Effect of Cache Parameters on Performance

<table>
<thead>
<tr>
<th></th>
<th>Larger capacity cache</th>
<th>Higher associativity cache</th>
<th>Larger block size cache *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compulsory misses</td>
<td>=</td>
<td>=</td>
<td>↓</td>
</tr>
<tr>
<td>Capacity misses</td>
<td>↓</td>
<td>=</td>
<td>↓</td>
</tr>
<tr>
<td>Conflict misses</td>
<td>↓</td>
<td>↓</td>
<td>?</td>
</tr>
<tr>
<td>Hit latency</td>
<td>↑</td>
<td>↑</td>
<td>=</td>
</tr>
<tr>
<td>Miss latency</td>
<td>=</td>
<td>=</td>
<td>↑↓</td>
</tr>
</tbody>
</table>

* Assume substantial spatial locality
Block-level Optimizations

- Tags are too large, i.e., too much overhead
  - Simple solution: Larger blocks, but miss penalty could be large.

- Sub-block placement (aka sector cache)
  - A valid bit added to units smaller than the full block, called sub-blocks
  - Only read a sub-block on a miss
  - *If a tag matches, is the word in the cache?*

\[
\begin{array}{cccc}
100 & 1 & 1 & 1 & 1 \\
300 & 1 & 1 & 0 & 0 \\
204 & 0 & 1 & 0 & 1 \\
\end{array}
\]
Replacement Policy

Which block from a set should be evicted?

- Random

- Least Recently Used (LRU)
  - LRU cache state must be updated on every access
  - true implementation only feasible for small sets (2-way)
  - pseudo-LRU binary tree was often used for 4-8 way

- First In, First Out (FIFO) a.k.a. Round-Robin
  - used in highly associative caches

- Not Least Recently Used (NLRU)
  - FIFO with exception for most recently used block or blocks

- One-bit LRU
  - Each way represented by a bit. Set on use, replace first unused.
Multiple replacement policies

Use the best replacement policy for a program

How do we decide which policy to use?

Cache

Sets

Miss

0: Policy A Missed
1: Policy B Missed

+1 -1

Counter

>0

0: Policy A
1: Policy B
Multilevel Caches

- A memory cannot be large and fast
- Add level of cache to reduce miss penalty
  - Each level can have longer latency than level above
  - So, increase sizes of cache at each level

![Cache Diagram]

Metrics:

Local miss rate = misses in cache / accesses to cache
Global miss rate = misses in cache / CPU memory accesses
Misses per instruction = misses in cache / number of instructions
Inclusion Policy

• **Inclusive multilevel cache:**
  - Inner cache holds copies of data in outer cache
  - External access need only check outer cache
  - Most common case

• **Exclusive multilevel caches:**
  - Inner cache may hold data not in outer cache
  - Swap lines between inner/outer caches on miss
  - Used in AMD Athlon with 64KB primary and 256KB secondary cache

Why choose one type or the other?
Victim Caches (HP 7200)

Victim cache is a small associative back up cache, added to a direct mapped cache, which holds recently evicted lines.
- First look up in direct mapped cache
- If miss, look in victim cache
- If hit in victim cache, swap hit line with line now evicted from L1
- If miss in victim cache, L1 victim -> VC, VC victim->?

Fast hit time of direct mapped but with reduced conflict misses
Typical memory hierarchies

(a) Memory hierarchy for server
- CPU
- L1 Cache
- L2 Cache
- L3 Cache
- Memory
- I/O bus
- Disk storage

Register reference
Size: 1000 bytes
Speed: 300 ps

Level 1 Cache reference
Size: 64 KB
Speed: 1 ns

Level 2 Cache reference
Size: 256 KB
Speed: 3–10 ns

Level 3 Cache reference
Size: 2–4 MB
Speed: 10–20 ns

Memory reference
Size: 4–16 GB
Speed: 50–100 ns

Disk memory reference
Size: 4–16 TB
Speed: 5–10 ms

(b) Memory hierarchy for a personal mobile device
- CPU
- L1 Cache
- L2 Cache
- Memory
- Storage

Register reference
Size: 500 bytes
Speed: 500 ps

Level 1 Cache reference
Size: 64 KB
Speed: 2 ns

Level 2 Cache reference
Size: 256 KB
Speed: 10–20 ns

Memory reference
Size: 256–512 MB
Speed: 50–100 ns

FLASH memory reference
Size: 4–8 GB
Speed: 25–50 us
HBM DRAM or MCDRAM

Source: AMD
Mixed technology caching (Intel Knights Landing)

CPU
L2

MCDRAM (as cache)

DDR

MCDRAM (as mem)

DDR

MCDRAM (as cache)
MCDRAM (as mem)
Thank you!

Next lecture – virtual memory