Review
- Time (seconds/program) is performance measure
  \[ \text{Performance} = \frac{\text{Instructions}}{\text{Clock cycles}} \times \frac{\text{Clock Cycle}}{\text{Program}} \]
- Benchmarks stand in for real workloads to as a standardized measure of relative performance
- Power of increasing concern, and being added to benchmarks
- Time measurement via clock cycles, machine specific
- Profiling tools as way to see where spending time in your program

New-School Machine Structures (It’s a bit more complicated!)
- Parallel Requests
  - Assigned to computer
  - e.g., Search “Katz”
- Parallel Threads
  - Assigned to core
  - e.g., Lookup, Ads
- Parallel Instructions
  - >1 instruction @ one time
  - e.g., 5 pipelined instructions
- Parallel Data
  - >1 data item @ one time
  - e.g., Add of 4 pairs of words
- Hardware descriptions
  - All gates @ one time

Library Analogy
- Writing a report based on books on reserve
  - E.g., works of J.D. Salinger
- Go to library to get reserved book and place on desk in library
- If need more, check them out and keep on desk
  - But don’t return earlier books since might need them
- You hope this collection of ~10 books on desk enough to write report, despite 10 being only 0.00001% of books in UC Berkeley libraries

Big Idea: Locality
- **Temporal Locality** (locality in time)
  - Go back to same book on desktop multiple times
  - If a memory location is referenced then it will tend to be referenced again soon
- **Spatial Locality** (locality in space)
  - When go to bookshelf, pick up multiple books on J.D. Salinger since library stores related books together
  - If a memory location is referenced, the locations with nearby addresses will tend to be referenced soon
Principle of Locality

• **Principle of Locality:** Programs access small portion of address space at any instant of time
• What program structures lead to temporal and spatial locality in code?
• In data?

How does hardware exploit principle of locality?

• Offer a hierarchy of memories where
  – closest to processor is fastest
  (and most expensive per bit so smallest)
  – furthest from processor is largest
  (and least expensive per bit so slowest)
• Goal is to create illusion of memory almost as fast as fastest memory and almost as large as biggest memory of the hierarchy

Big Idea: Memory Hierarchy

As we move to deeper levels the latency goes up and price per bit goes down. Why?

Cache Concept

• Processor and memory speed mismatch leads us to add a new level: a memory *cache*
• Implemented with same integrated circuit processing technology as processor, integrated on-chip: faster but more expensive than DRAM memory
• **Cache is a copy of a subset of main memory**
• Modern processors have separate caches for instructions and data, as well as several levels of caches implemented in different sizes
• As a pun, often use $\$ (“cash”) to abbreviate cache, e.g. $D$ = Data Cache, $I$ = Instruction Cache

Memory Hierarchy Technologies

• Caches use SRAM (Static RAM) for speed and technology compatibility
  – Fast (typical access times of 0.5 to 2.5 ns)
  – Low density (6 transistor cells), higher power, expensive ($\$2000$ to $\$4000$ per GB in 2011)
  – Static: content will last as long as power is on
• Main memory uses DRAM (Dynamic RAM) for size (density)
  – Slower (typical access times of 50 to 70 ns)
  – High density (1 transistor cells), lower power, cheaper ($\$20$ to $\$40$ per GB in 2011)
  – Dynamic: needs to be “refreshed” regularly (~ every 8 ms)
• Consumes 1% to 2% of the active cycles of the DRAM

Characteristics of the Memory Hierarchy

<table>
<thead>
<tr>
<th>Processor</th>
<th>Processor</th>
<th>Processor</th>
<th>Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-byte (word)</td>
<td>16-byte (block)</td>
<td>64-kbyte (page)</td>
<td>1-Mbyte (page)</td>
</tr>
<tr>
<td>8-32 bytes</td>
<td>512 bytes (block)</td>
<td>4,096+ bytes (page)</td>
<td>8-32 bytes</td>
</tr>
<tr>
<td>Increasing distance from the processor in access time</td>
<td>Increasing distance from the processor in access time</td>
<td>Increasing distance from the processor in access time</td>
<td>Increasing distance from the processor in access time</td>
</tr>
</tbody>
</table>

(Relative) size of the memory at each level

Inclusive—what is in L1$S$ is a subset of what is in L2$S$ is a subset of what is in MM that is a subset of is in SM
How is the Hierarchy Managed?

- registers ↔ memory
  - By compiler (or assembly level programmer)
- cache ↔ main memory
  - By the cache controller hardware
- main memory ↔ disks (secondary storage)
  - By the operating system (virtual memory)
  - (Talk about later in the semester)
  - Virtual to physical address mapping assisted by the hardware (TLB)
  - By the programmer (files)

Midterm in 2 weeks:
No Homework this week!

Project #2, Part 2 Due Sunday @ 11:59:59

Lab #6 posted

Spring 2011 - Lecture #11

Typical Memory Hierarchy

Review so far

- Wanted: effect of a large, cheap, fast memory
- Approach: Memory Hierarchy
  - Successively lower levels contain "most used" data from next higher level
  - Exploits temporal & spatial locality
- Memory hierarchy follows 2 Design Principles:
  - Smaller is faster and Do common the case fast

Agenda

- Memory Hierarchy Overview
- Administrivia
- Caches
- Direct Mapped Cache
- Technology Break
  - (Cache Performance: if time permits)

Spring 2011 - Lecture #11

Administrivia

- Everyone off wait lists
- Lab #6 posted
- Project #2, Part 2 Due Sunday @ 11:59:59
- No Homework this week!
- Midterm in 2 weeks:
  - Exam: Tu, Mar 8, 6-9 PM, 145/155 Dwinelle
    - Split: A-Law in 145, B-2 in 155
    - No discussion during exam week; no lecture that day
  - TA Review: Su, Mar 6, 2-5 PM, 2050 VLSB
  - Small number of special consideration cases, due to class conflicts, etc.—contact Dave or Randy

Spring 2011 - Lecture #11

Project 2, Part 1

- 12 new tests
- 57 passed all tests
  - Max score 15
- 218 fail 4 to 6 tests
  - Most code does not properly sign-extend immediate field
- 4 failed initial 2 tests
- 3 didn’t compile

Spring 2011 - Lecture #11
61C in the News

“South Korean homes now have greater Internet access than we do,” President Obama said in his State of the Union address. Last week, Mr. Obama unveiled an $18.7 billion broadband spending program. (Use WiMax to beam broadband from where Internet providers’ systems currently end to rural areas.)

By the end of 2012, South Korea intends to connect every home in the country to the Internet at one gigabit per second. That would be a tenfold increase from the already blazing national standard and more than 200 times as fast as the average household setup in the United States.

What Conference?

- Will be held in charming San Francisco April 3-5?
- Is sponsored by Google, Intel, Microsoft, Cisco, NetAp, ...
- Includes a past National Academy of Engineering President, past IBM Academy of Technology Chair, past ACM President, past Computing Research Association Chair, members of NAE, Technology Review’s Young Innovators (TR35), and Google’s Executive VP of Research & Engineering?
- Has attendance that averages 50% female, 40% African American, and 30% Hispanic?

Cache Basics: Direct Mapped

- Direct mapped
  - Each memory block is mapped to exactly one block in the cache
  - Many lower level blocks share a given cache block
  - Block is also called a line
- Address mapping:
  - (block address) modulo (# of blocks in the cache)
  - Tag associated with each cache block containing the address information (the upper portion of the address) required to identify the block (to answer Q1)
  - (Later in semester we’ll cover alternatives to direct mapped)
Spring 2011 - Lecture #11

**Caching: A Simple First Example**

<table>
<thead>
<tr>
<th>Cache</th>
<th>Main Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>Valid</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
</tr>
</tbody>
</table>

Q: Is the mem block in cache?

Compare the cache tag to the given memory address bits to tell if the memory block is in the cache.

```
(block address) modulo (# of blocks in the cache)
```

**Direct Mapped Cache Example**

- One word blocks, cache size = 1K words (or 4KB)

```
Valid bit ensures something useful in cache for this index
```

**Direct Mapped Cache**

- Consider the main memory word reference string
- Start with an empty cache - all blocks initially marked as not valid

<table>
<thead>
<tr>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0001 0010 0011 0100 0011 0100 1111</td>
</tr>
</tbody>
</table>

```
0 miss
```

• 1 requests, 1 miss
Direct Mapped Cache
- Consider the main memory word reference string
- Start with an empty cache - all blocks initially marked as not valid

Multiword Block Direct Mapped Cache
- Four words/block, cache size = 1K words

Taking Advantage of Spatial Locality
- Let cache block hold more than one word
- Start with an empty cache - all blocks initially marked as not valid
Miss Rate vs Block Size vs Cache Size

- Miss rate goes up if the block size becomes a significant fraction of the cache size because the number of blocks that can be held in the same size cache is smaller (increasing capacity misses).

Impacts of Cache Performance

- Relative $S$ penalty increases as processor performance improves (faster clock rate and/or lower CPI).
  - Memory speed unlikely to improve as fast as processor cycle time. When calculating $CPI_{mem}$, cache miss penalty is measured in processor clock cycles needed to handle a miss.
  - Lower the CPI$_{ideal}$, more pronounced impact of stalls.
- Processor with a CPI$_{ideal}$ of 2, a 100 cycle miss penalty, 36% load/store instr's, and 2% IS and 4% DS miss rates.
  - Memory-stall cycles $= 2\times 100 + 36\times 4\% \times 100 = 3.44$
  - So $CPI_{mem} = 2 + 3.44 = 5.44$
  - More than twice the CPI$_{ideal}$!
- What if the CPI$_{ideal}$ is reduced to 1?
- What if the DS miss rate went up by 1%?

Measuring Cache Performance

- Assuming cache hit costs are included as part of the normal CPU execution cycle, then:
  \[ CPU\ time = IC \times CPI \times CC \]

- A simple model for Memory-stall cycles:
  \[ Memory\-stall\ cycles = accesses/program \times miss\ rate \times miss\ penalty \]

- Will talk about writes and write misses next lecture, where it’s a little more complicated.

Average Memory Access Time (AMAT)

- Average Memory Access Time (AMAT) is the average to access memory considering both hits and misses:
  \[ AMAT = Time\ for\ a\ hit + Miss\ rate \times Miss\ penalty \]

- What is the AMAT for a processor with a 200 psec clock, a miss penalty of 50 clock cycles, a miss rate of 0.02 misses per instruction and a cache access time of 1 clock cycle?

- Potential impact of much larger cache on AMAT?
  1) Lower Miss rate
  2) Longer Access time (Hit time): smaller is faster.

- Increase in hit time will likely add another stage to the pipeline:
  - At some point, increase in hit time for a larger cache may overcome the improvement in hit rate, yielding a decrease in performance.

Review so far

- Principle of Locality for Libraries / Computer Memory
- Hierarchy of Memories (speed/size/cost per bit) to Exploit Locality
- Cache – copy of data lower level in memory hierarchy
- Direct Mapped to find block in cache using Tag field and Valid bit for Hit
- Larger caches reduce Miss rate via Temporal and Spatial Locality, but can increase Hit time.
- Larger blocks to reduces Miss rate via Spatial Locality, but increase Miss penalty
- AMAT helps balance Hit time, Miss rate, Miss penalty