Introduction

- Programmers want unlimited amounts of memory with low latency
- Fast memory technology is more expensive per bit than slower memory
- Solution: organize memory system into a hierarchy
  - Entire addressable memory space available in largest, slowest memory
  - Incrementally smaller and faster memories, each containing a subset of the memory below it, proceed in steps up toward the processor
- Temporal and spatial locality insures that nearly all references can be found in smaller memories
  - Gives the allusion of a large, fast memory being presented to the processor
Memory Hierarchy

Introduction

Memory Performance Gap

Introduction
Memory Hierarchy Design

- Memory hierarchy design becomes more crucial with recent multi-core processors:
  - Aggregate peak bandwidth grows with # cores:
    - Intel Core i7 can generate two references per core per clock
    - Four cores and 3.2 GHz clock
      - 25.6 billion 64-bit data references/second +
      - 12.8 billion 128-bit instruction references
      - = 409.6 GB/s!
  - DRAM bandwidth is only 6% of this (25 GB/s)
  - Requires:
    - Multi-port, pipelined caches
    - Two levels of cache per core
    - Shared third-level cache on chip

Performance and Power

- High-end microprocessors have >10 MB on-chip cache
  - Consumes large amount of area and power budget
Memory Hierarchy Basics

- When a word is not found in the cache, a *miss* occurs:
  - Fetch word from lower level in hierarchy, requiring a higher latency reference
  - Lower level may be another cache or the main memory
  - Also fetch the other words contained within the *block*
    - Takes advantage of spatial locality
  - Place block into cache in any location within its *set*, determined by address
    - block address MOD number of sets

**Introduction**

- \( n \) sets => *n*-way set associative
  - *Direct-mapped cache* => one block per set
  - *Fully associative* => one set

- Writing to cache: two strategies
  - *Write-through*
    - Immediately update lower levels of hierarchy
  - *Write-back*
    - Only update lower levels of hierarchy when an updated block is replaced
  - Both strategies use *write buffer* to make writes asynchronous
Memory Hierarchy Basics

- Miss rate
  - Fraction of cache access that result in a miss

- Causes of misses
  - Compulsory
    - First reference to a block
  - Capacity
    - Blocks discarded and later retrieved
  - Conflict
    - Program makes repeated references to multiple addresses from different blocks that map to the same location in the cache

[Miss rate equation]

Introduction

- Note that speculative and multithreaded processors may execute other instructions during a miss
  - Reduces performance impact of misses
## Memory Hierarchy Basics

- Six basic cache optimizations:
  - Larger block size
    - Reduces compulsory misses
    - Increases capacity and conflict misses, increases miss penalty
  - Larger total cache capacity to reduce miss rate
    - Increases hit time, increases power consumption
  - Higher associativity
    - Reduces conflict misses
    - Increases hit time, increases power consumption
  - Higher number of cache levels
    - Reduces overall memory access time
  - Giving priority to read misses over writes
    - Reduces miss penalty
  - Avoiding address translation in cache indexing
    - Reduces hit time

---

## Ten Advanced Optimizations

- Small and simple first level caches
  - Critical timing path:
    - Addressing tag memory, then
    - Comparing tags, then
    - Selecting correct set
  - Direct-mapped caches can overlap tag compare and transmission of data
  - Lower associativity reduces power because fewer cache lines are accessed
L1 Size and Associativity

Access time vs. size and associativity

Energy per read vs. size and associativity
Way Prediction

- To improve hit time, predict the way to pre-set mux
  - Mis-prediction gives longer hit time
  - Prediction accuracy
    - > 90% for two-way
    - > 80% for four-way
    - L-cache has better accuracy than D-cache
  - First used on MIPS R10000 in mid-90s
  - Used on ARM Cortex-A8
- Extend to predict block as well
  - “Way selection”
  - Increases mis-prediction penalty

Pipelining Cache

- Pipeline cache access to improve bandwidth
  - Examples:
    - Pentium: 1 cycle
    - Pentium Pro – Pentium III: 2 cycles
    - Pentium 4 – Core i7: 4 cycles
- Increases branch mis-prediction penalty
- Makes it easier to increase associativity
Nonblocking Caches

- Allow hits before previous misses complete
  - “Hit under miss”
  - “Hit under multiple miss”
- L2 must support this
- In general, processors can hide L1 miss penalty but not L2 miss penalty

Multibanked Caches

- Organize cache as independent banks to support simultaneous access
  - ARM Cortex-A8 supports 1-4 banks for L2
  - Intel i7 supports 4 banks for L1 and 8 banks for L2
- Interleave banks according to block address

Figure 2.6 Four-way interleaved cache banks using block addressing. Assuming 64 bytes per block, each of these addresses would be multiplied by 64 to get byte addressing.
Critical Word First, Early Restart

- Critical word first
  - Request missed word from memory first
  - Send it to the processor as soon as it arrives
- Early restart
  - Request words in normal order
  - Send missed work to the processor as soon as it arrives

Effectiveness of these strategies depends on block size and likelihood of another access to the portion of the block that has not yet been fetched.

Merging Write Buffer

- When storing to a block that is already pending in the write buffer, update write buffer
- Reduces stalls due to full write buffer
- Do not apply to I/O addresses
Compiler Optimizations

- Loop Interchange
  - Swap nested loops to access memory in sequential order

- Blocking
  - Instead of accessing entire rows or columns, subdivide matrices into blocks
  - Requires more memory accesses but improves locality of accesses

Hardware Prefetching

- Fetch two blocks on miss (include next sequential block)

Pentium 4 Pre-fetching
Compiler Prefetching

- Insert prefetch instructions before data is needed
- Non-faulting: prefetch doesn’t cause exceptions
- Register prefetch
  - Loads data into register
- Cache prefetch
  - Loads data into cache
- Combine with loop unrolling and software pipelining

Summary

<table>
<thead>
<tr>
<th>Technique</th>
<th>Hit time</th>
<th>Bandwidth</th>
<th>Miss penalty</th>
<th>Miss rate</th>
<th>Power consumption</th>
<th>Hardware complexity</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small and simple caches</td>
<td>+</td>
<td>–</td>
<td>+</td>
<td>0</td>
<td>Trivial, widely used</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Way-predicting caches</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>1</td>
<td>Used in Pentium 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pipelined cache access</td>
<td>–</td>
<td>+</td>
<td></td>
<td>1</td>
<td>Widely used</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nonlocking caches</td>
<td>+</td>
<td>–</td>
<td></td>
<td>3</td>
<td>Widespread use</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Buffered caches</td>
<td>+</td>
<td>–</td>
<td>+</td>
<td>1</td>
<td>Used in L2 of both i7 and Cortex-A9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Critical word first and early return</td>
<td>+</td>
<td>+</td>
<td></td>
<td>2</td>
<td>Widely used</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory write buffer</td>
<td>+</td>
<td>+</td>
<td></td>
<td>1</td>
<td>Widespread write through</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complex techniques to reduce cache misses</td>
<td>+</td>
<td>–</td>
<td></td>
<td>0</td>
<td>Software is a challenge, but most compilers handle common linear algebra calculations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware prefetching of instructions and data</td>
<td>+</td>
<td>–</td>
<td></td>
<td>2 instr., 3 data</td>
<td>Most provide prefetch instructions; modern high-end processors also automatically prefetch in hardware</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complex-controlled prefetching</td>
<td>+</td>
<td>+</td>
<td></td>
<td>3</td>
<td>Needs non-blocking cache, possibly instruction overhead in many CPUs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Memory Technology

- Performance metrics
  - Latency is concern of cache
  - Bandwidth is concern of multiprocessors and I/O
  - Access time
    - Time between read request and when desired word arrives
  - Cycle time
    - Minimum time between unrelated requests to memory

- DRAM used for main memory, SRAM used for cache

Memory Technology

- SRAM
  - Requires low power to retain bit
  - Requires 6 transistors/bit

- DRAM
  - Must be re-written after being read
  - Must also be periodically refreshed
    - Every ~ 8 ms
    - Each row can be refreshed simultaneously
  - One transistor/bit
  - Address lines are multiplexed:
    - Upper half of address: row access strobe (RAS)
    - Lower half of address: column access strobe (CAS)
Memory Technology

- **Amdahl:**
  - Memory capacity should grow linearly with processor speed
  - Unfortunately, memory capacity and speed has not kept pace with processors

- **Some optimizations:**
  - Multiple accesses to same row
  - Synchronous DRAM
    - Added clock to DRAM interface
    - Burst mode with critical word first
  - Wider interfaces
  - Double data rate (DDR)
  - Multiple banks on each DRAM device

<table>
<thead>
<tr>
<th>Production year</th>
<th>Chip size</th>
<th>DRAM Type</th>
<th>Fastest DRAM (ns)</th>
<th>Slowest DRAM (ns)</th>
<th>Column access strobe (CAS) / data transfer time (ns)</th>
<th>Cycle time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1980</td>
<td>64K bit</td>
<td>DRAM</td>
<td>180</td>
<td>150</td>
<td>75</td>
<td>250</td>
</tr>
<tr>
<td>1983</td>
<td>256K bit</td>
<td>DRAM</td>
<td>150</td>
<td>120</td>
<td>50</td>
<td>220</td>
</tr>
<tr>
<td>1986</td>
<td>1M bit</td>
<td>DRAM</td>
<td>120</td>
<td>100</td>
<td>25</td>
<td>190</td>
</tr>
<tr>
<td>1989</td>
<td>4M bit</td>
<td>DRAM</td>
<td>100</td>
<td>80</td>
<td>20</td>
<td>165</td>
</tr>
<tr>
<td>1992</td>
<td>16M bit</td>
<td>DRAM</td>
<td>80</td>
<td>60</td>
<td>15</td>
<td>120</td>
</tr>
<tr>
<td>1996</td>
<td>64M bit</td>
<td>SDRAM</td>
<td>70</td>
<td>50</td>
<td>12</td>
<td>110</td>
</tr>
<tr>
<td>1998</td>
<td>128M bit</td>
<td>SDRAM</td>
<td>70</td>
<td>50</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>2000</td>
<td>256M bit</td>
<td>DDR1</td>
<td>65</td>
<td>45</td>
<td>7</td>
<td>90</td>
</tr>
<tr>
<td>2002</td>
<td>512M bit</td>
<td>DDR1</td>
<td>60</td>
<td>40</td>
<td>5</td>
<td>80</td>
</tr>
<tr>
<td>2004</td>
<td>1G bit</td>
<td>DDR2</td>
<td>55</td>
<td>35</td>
<td>5</td>
<td>70</td>
</tr>
<tr>
<td>2006</td>
<td>2G bit</td>
<td>DDR2</td>
<td>50</td>
<td>30</td>
<td>2.5</td>
<td>60</td>
</tr>
<tr>
<td>2010</td>
<td>4G bit</td>
<td>DDR3</td>
<td>36</td>
<td>28</td>
<td>1</td>
<td>37</td>
</tr>
</tbody>
</table>
| 2012            | 8G bit    | DDR3      | 30                | 24               | 0.5                                              | 31             

*Figure 2.18* Times of fast and slow DRAMs vary with each generation. (Cycle time is defined on page 95.) Performance improvement of row access time is about 5% per year. The improvement by a factor of 2 in column access in 1986 accompanied the switch from NMOS DRAMs to CMOS DRAMs. The introduction of various burst transfer modes in the mid-1990s and SDRAMs in the late 1990s has significantly complicated the calculation of access time for blocks of data; we discuss this later in this section when we talk about SDRAM access time and power. The DDR4 designs are due for introduction in mid-to late 2013. We discuss these various forms of DRAMs in the next few pages.
**Memory Optimizations**

<table>
<thead>
<tr>
<th>Standard</th>
<th>Clock rate (MHz)</th>
<th>M transfers per second</th>
<th>DRAM name</th>
<th>MB/sec /DIMM</th>
<th>DIMM name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR</td>
<td>133</td>
<td>266</td>
<td>DDR-266</td>
<td>2128</td>
<td>PC2100</td>
</tr>
<tr>
<td>DDR</td>
<td>150</td>
<td>300</td>
<td>DDR-300</td>
<td>2400</td>
<td>PC2000</td>
</tr>
<tr>
<td>DDR</td>
<td>200</td>
<td>400</td>
<td>DDR-400</td>
<td>3200</td>
<td>PC3200</td>
</tr>
<tr>
<td>DDR2</td>
<td>266</td>
<td>533</td>
<td>DDR2-533</td>
<td>4264</td>
<td>PC4300</td>
</tr>
<tr>
<td>DDR2</td>
<td>333</td>
<td>667</td>
<td>DDR2-667</td>
<td>5336</td>
<td>PC5300</td>
</tr>
<tr>
<td>DDR2</td>
<td>400</td>
<td>800</td>
<td>DDR2-800</td>
<td>6400</td>
<td>PC6400</td>
</tr>
<tr>
<td>DDR3</td>
<td>533</td>
<td>1066</td>
<td>DDR3-1066</td>
<td>8528</td>
<td>PC8500</td>
</tr>
<tr>
<td>DDR3</td>
<td>666</td>
<td>1333</td>
<td>DDR3-1333</td>
<td>10,664</td>
<td>PC10700</td>
</tr>
<tr>
<td>DDR3</td>
<td>800</td>
<td>1600</td>
<td>DDR3-1600</td>
<td>12,800</td>
<td>PC12800</td>
</tr>
<tr>
<td>DDR4</td>
<td>1066–1600</td>
<td>2133–3200</td>
<td>DDR4-3200</td>
<td>17,056–25,600</td>
<td>PC25600</td>
</tr>
</tbody>
</table>

Figure 2.14 Clock rates, bandwidth, and names of DDR DRAMS and DIMMs in 2010. Note the numerical relationship between the columns. The third column is twice the second, and the fourth uses the number from the third column in the name of the DRAM chip. The fifth column is eight times the third column, and a rounded version of this number is used in the name of the DIMM. Although not shown in this figure, DDRs also specify latency in clock cycles as four numbers, which are specified by the DDR standard. For example, DDR3-2000 CL 9 has latencies of 9-9-9-28. What does this mean? With a 1 ns clock (clock cycle is one-half the transfer rate), this indicates 9 ns for row to columns address (RAS time), 9 ns for row access data (CAS time), and a minimum read time of 28 ns. Closing the row takes 9 ns for precharge but happens only when the reads from that row are finished. In burst mode, transfers occur on every clock on both edges, when the first RAS and CAS times have elapsed. Furthermore, the precharge in not needed until the entire row is read. DDR4 will be produced in 2012 and is expected to reach clock rates of 1600 MHz in 2014, when DDR5 is expected to take over. The exercises explore these details further.

**Memory Optimizations**

- **DDR:**
  - **DDR2**
    - Lower power (2.5 V → 1.8 V)
    - Higher clock rates (266 MHz, 333 MHz, 400 MHz)
  - **DDR3**
    - 1.5 V
    - 800 MHz
  - **DDR4**
    - 1-1.2 V
    - 1600 MHz

- **GDDR5** is graphics memory based on DDR3
Memory Optimizations

- Graphics memory:
  - Achieve 2-5 X bandwidth per DRAM vs. DDR3
    - Wider interfaces (32 vs. 16 bit)
    - Higher clock rate
      - Possible because they are attached via soldering instead of socketed DIMM modules

- Reducing power in SDRAMs:
  - Lower voltage
  - Low power mode (ignores clock, continues to refresh)

Memory Power Consumption

- Graph showing power consumption in mW for different modes:
  - Low power mode
  - Typical usage
  - Fully active

- Legend:
  - Read, write, terminate power
  - Activate power
  - Background power
Flash Memory

- Type of EEPROM
- Must be erased (in blocks) before being overwritten
- Non volatile
- Limited number of write cycles
- Cheaper than SDRAM, more expensive than disk
- Slower than SRAM, faster than disk

Memory Dependability

- Memory is susceptible to cosmic rays
- **Soft errors**: dynamic errors
  - Detected and fixed by error correcting codes (ECC)
- **Hard errors**: permanent errors
  - Use sparse rows to replace defective rows
- **Chipkill**: a RAID-like error recovery technique
Virtual Memory

- Protection via virtual memory
  - Keeps processes in their own memory space

- Role of architecture:
  - Provide user mode and supervisor mode
  - Protect certain aspects of CPU state
  - Provide mechanisms for switching between user mode and supervisor mode
  - Provide mechanisms to limit memory accesses
  - Provide TLB to translate addresses

Virtual Machines

- Supports isolation and security
- Sharing a computer among many unrelated users
- Enabled by raw speed of processors, making the overhead more acceptable

- Allows different ISAs and operating systems to be presented to user programs
  - “System Virtual Machines”
  - SVM software is called “virtual machine monitor” or “hypervisor”
  - Individual virtual machines run under the monitor are called “guest VMs”
### Impact of VMs on Virtual Memory

- Each guest OS maintains its own set of page tables
  - VMM adds a level of memory between physical and virtual memory called “real memory”
  - VMM maintains shadow page table that maps guest virtual addresses to physical addresses
    - Requires VMM to detect guest's changes to its own page table
    - Occurs naturally if accessing the page table pointer is a privileged operation