Recap

- We have encountered a few concepts
  - Program locality
  - TLB and cache
  - Virtual memory

- But what they are and why/how they work?
  - This week’s topics
Today’s Topics

- Memory/storage technologies and trends
  - Memory wall!

- Locality of reference to the rescue
  - Caching in the memory hierarchy
Random-Access Memory (RAM)

- Key features
  - RAM is traditionally packaged as a chip.
  - Basic storage unit is normally a cell (one bit per cell).
  - Multiple RAM chips form a memory.

- Static RAM (SRAM)
  - Each cell stores a bit with a four or six-transistor circuit.
  - Retains value indefinitely, as long as it is kept powered.
  - Relatively insensitive to electrical noise (EMI), radiation, etc.
  - Faster and more expensive than DRAM.

- Dynamic RAM (DRAM)
  - Each cell stores bit with a capacitor. One transistor is used for access.
  - Value must be refreshed every 10-100 ms.
  - More sensitive to disturbances (EMI, radiation,…) than SRAM.
  - Slower and cheaper than SRAM.
## SRAM vs DRAM Summary

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM</td>
<td>4 or 6</td>
<td>1X</td>
<td>No</td>
<td>Maybe</td>
<td>100x</td>
<td>Cache memories</td>
</tr>
<tr>
<td>DRAM</td>
<td>1</td>
<td>10X</td>
<td>Yes</td>
<td>Yes</td>
<td>1X</td>
<td>Main memories, frame buffers</td>
</tr>
</tbody>
</table>

CS 153 – Lecture 18 – Memory Hierarchy
Conventional DRAM Organization

- d x w DRAM:
  - dw total bits organized as d supercells of size w bits

```
<table>
<thead>
<tr>
<th>cols</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>
```

```
Memory controller
```

```
16 x 8 DRAM chip
```

```
Internal row buffer
```

```
 addr

0 bits /

2 bits /

8 bits /

data
```
Reading DRAM Supercell (2,1)

Step 1(a): Row access strobe (RAS) selects row 2.
Step 1(b): Row 2 copied from DRAM array to row buffer.
Reading DRAM Supercell (2,1)

Step 2(a): Column access strobe (CAS) selects column 1.

Step 2(b): Supercell (2,1) copied from buffer to data lines, and eventually back to the CPU.
Memory Modules

64 MB memory module consisting of eight 8Mx8 DRAMs

addr (row = i, col = j)

Memory controller

64-bit doubleword at main memory address A

64-bit doubleword
Enhanced DRAMs

- Basic DRAM cell has not changed since its invention in 1966.
  - Commercialized by Intel in 1970.

- DRAM cores with better interface logic and faster I/O:
  - Synchronous DRAM (SDRAM)
    - Uses a conventional clock signal instead of asynchronous control
    - Allows reuse of the row addresses (e.g., RAS, CAS, CAS, CAS)
  - Double data-rate synchronous DRAM (DDR SDRAM)
    - Double edge clocking sends two bits per cycle per pin
    - Different types distinguished by size of small prefetch buffer:
      - DDR (2 bits), DDR2 (4 bits), DDR4 (8 bits)
    - By 2010, standard for most server and desktop systems
    - Intel Core i7 supports only DDR3 SDRAM
Nonvolatile Memories

- DRAM and SRAM are volatile – lose info without power

- Nonvolatile memories (NVMs) retain value
  - Read-only memory (ROM): programmed during production
  - Programmable ROM (PROM): can be programmed once
  - Eraseable PROM (EPROM): can be bulk erased (UV, X-Ray)
  - Electrically eraseable PROM (EEPROM): electronic erase
  - Flash memory: EEPROMs with partial (sector) erase capability
    » Wears out after about 100,000 erasings.
  - Phase Change Memories (PCMs): also wear out
  - Many exciting NVMs at various stages of development
NVM Uses

- Firmware programs stored in a ROM (BIOS, controllers for disks, network cards, graphics accelerators, security subsystems, ...)

- Solid state disks (replace rotating disks in thumb drives, smart phones, mp3 players, tablets, laptops, ...)

- Caches in high end systems

- Getting better -- many expect Universal memory to come
  - i.e., large replace both DRAM and disk drives
The CPU-Memory Gap

The gap widens between DRAM, disk, and CPU speeds.

- **Disk seek time**
- **Flash SSD access time**
- **DRAM access time**
- **SRAM access time**
- **CPU cycle time**
- **Effective CPU cycle time**
The Price-Speed Gap

- **SRAM**
  - Latency: 0.5-2.5 ns, cost: ~$5000 per GB

- **DRAM**
  - Latency: 50-70 ns, cost: ~$20 - $50 per GB

- **SSD/NVM**
  - Latency: 70-150 ns, cost: ~$4 - $12 per GB

- **Magnetic disk**
  - Latency: 5-20 ms, cost: ~$0.02 - $2 per GB
Locality to the Rescue!

- The key to bridging this CPU-Memory gap is a fundamental property of computer programs known as locality.
Locality

- **Principle of Locality**: Programs tend to use data and instructions with addresses near or equal to those they have used recently.

- **Temporal locality**: Recently referenced items are likely to be referenced again in the near future.

- **Spatial locality**: Items with nearby addresses tend to be referenced close together in time.
Memory Hierarchies

- Some fundamental and enduring properties of hardware and software:
  - Fast storage technologies cost more per byte, have less capacity, and require more power (heat!).
  - The gap between CPU and main memory speed is widening.
  - Well-written programs tend to exhibit good locality.

- These fundamental properties complement each other beautifully.

- They suggest an approach for organizing memory and storage systems known as a memory hierarchy.
An Example Memory Hierarchy

- **L0:** Registers
  - CPU registers hold words retrieved from L1 cache

- **L1:** L1 cache (SRAM)
  - L1 cache holds cache lines retrieved from L2 cache

- **L2:** L2 cache (SRAM)
  - L2 cache holds cache lines retrieved from main memory

- **L3:** Main memory (DRAM)
  - Main memory holds disk blocks retrieved from local disks

- **L4:** Local secondary storage (local disks)
  - Local disks hold files retrieved from disks on remote network servers

- **L5:** Remote secondary storage (tapes, distributed file systems, Web servers)

Smaller, faster, costlier per byte

Larger, slower, cheaper per byte
Memory hierarchy

- **Cache**: A smaller, faster storage device that acts as a staging area for a subset of the data in a larger, slower device.

- Fundamental idea of a memory hierarchy:
  For each layer, faster, smaller device caches larger, slower device

- Why do memory hierarchies work?
  Because of locality!
  - Hit fast memory much more frequently even though its smaller
  Thus, the storage at level k+1 can be slower (but larger and cheaper!)

- **Big Idea**: The memory hierarchy creates a large pool of storage that costs as much as the cheap storage near the bottom, but that serves data to programs at the rate of the fast storage near the top.
Virtual Memory (VM)

- Virtual memory is page with a new ingredient
  - Allow pages to be on disk
    - In a special partition (or file) called swap
  - i.e., memory as cache for disk

- Motivation?
  - Uses main memory efficiently
  - Use DRAM as a cache for the parts of a virtual address space

- Simplifies memory management
  - Each process gets the same uniform linear address space
  - With VM, this can be big!
# Examples of Caching in the Hierarchy

<table>
<thead>
<tr>
<th>Cache Type</th>
<th>What is Cached?</th>
<th>Where is it Cached?</th>
<th>Latency (cycles)</th>
<th>Managed By</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registers</td>
<td>4-8 bytes words</td>
<td>CPU core</td>
<td>0</td>
<td>Compiler</td>
</tr>
<tr>
<td>TLB</td>
<td>Address translations</td>
<td>On-Chip TLB</td>
<td>0</td>
<td>Hardware</td>
</tr>
<tr>
<td>L1 cache</td>
<td>64-bytes block</td>
<td>On-Chip L1</td>
<td>1</td>
<td>Hardware</td>
</tr>
<tr>
<td>L2 cache</td>
<td>64-bytes block</td>
<td>On/Off-Chip L2</td>
<td>10</td>
<td>Hardware</td>
</tr>
<tr>
<td>Virtual Memory</td>
<td>4-KB page</td>
<td>Main memory</td>
<td>100</td>
<td>Hardware + OS</td>
</tr>
<tr>
<td>Buffer cache</td>
<td>Parts of files</td>
<td>Main memory</td>
<td>100</td>
<td>OS</td>
</tr>
<tr>
<td>Disk cache</td>
<td>Disk sectors</td>
<td>Disk controller</td>
<td>100,000</td>
<td>Disk firmware</td>
</tr>
<tr>
<td>Network buffer cache</td>
<td>Parts of files</td>
<td>Local disk</td>
<td>10,000,000</td>
<td>AFS/NFS client</td>
</tr>
<tr>
<td>Browser cache</td>
<td>Web pages</td>
<td>Local disk</td>
<td>10,000,000</td>
<td>Web browser</td>
</tr>
<tr>
<td>Web cache</td>
<td>Web pages</td>
<td>Remote server disks</td>
<td>1,000,000,000</td>
<td>Web proxy server</td>
</tr>
</tbody>
</table>
Summary so far

- The speed gap between CPU, memory and mass storage continues to widen.

- Well-written programs exhibit a property called locality.

- Memory hierarchies based on caching close the gap by exploiting locality.
Next time

- More on locality and cache