Exploiting Memory Hierarchy (Chapter 5)

- **Users want large and fast memories (not possible)**
  - SRAM access times are 0.5 – 2.5 ns at cost of $500 to $1000 per Gbyte (in 2012)
  - DRAM access times are 50-70 ns at cost of $10 to $20 per Gbyte.
  - Flash access times are 5-50 us at cost of $0.75 to $1 per Gbyte
  - Disk access times are 5-20 ms at cost of $0.05 to $0.1 per Gbyte.

- **To give the illusion of large and fast memory**
  - build a memory hierarchy

![Memory Hierarchy Diagram]

**CPU**

- **L1 cache**
  - SRAM
  - Smaller
  - Faster
  - More expensive per byte

- **L2 cache**
  - SRAM

- **Main memory**
  - DRAM
  - Larger
  - Slower
  - Cheaper per byte

- **Hard disk / Flash**

---

**DRAM Vs SRAM**

- DRAM cell relies on a capacitor
  - Charged \(\rightarrow 1\), otherwise \(\rightarrow 0\)
- Slower but smaller
- Volatile (may loose charge over time)
- Need to be refreshed periodically
- Reads are destructive

- SRAM cell is built from 6 transistors
- Faster but larger
- Non-volatile – keeps its value
- Does not need to be refreshed
- Reads are not destructive
Arranging cells in a 2-D structure

Example:
- Assume a byte addressable memory that contains 512Bytes
- Need 9 bits to address one of the 512 bytes
- Bytes are arranged into an array of 64x8 bytes (to keep an aspect ratio of 1)
- A 9-bit address is translated into a row address (6 bits) and a byte address (3 bits)
- Assume row major ordering

<table>
<thead>
<tr>
<th>Byte address 164</th>
<th>Row address 20 + col address 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte address 010100100</th>
<th>Row address 010100 + col address 100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note:
- 20 = 164 / 8
- 4 = 164 mod 8

DRAM operation

- Each memory bank has a “row buffer”, which is non-volatile (registers)
- To read a byte (a similar process applies for writing):
  - The memory controller sends the row address of the byte
  - The entire row is read into the row buffer (the row is opened)
  - The memory controller sends the column address of the byte
  - The memory returns the byte to the controller (from the row buffer)

- The row buffer has to be written back (closed) because the read was destructive
  - **Closed row/page policy**: close the row after you read the data
  - **Open row/page policy**: close the row only when you want to open a new row – useful if you will read again from the same row.
Multiple banks per memory chip

- Assume 4 banks per chip, 512 bytes per bank → total 2048 bytes
- 2048 bytes → the address of each byte is 11 bits
  - 2 bits to select a bank
  - 6 bits to select a row
  - 3 bits to select a column
- Example of a possible interleaving

```
xxxxxx xx xxx
```

- Other ways to interleave
  ```
  xxxxxx xxx xx
  xx xxxxxx xxx
  ```

DRAM internals (not in textbook)

Example: a 128 Mbytes chip ($2^{27}$) → 4 banks, 32 Mbytes each
  a bank = 16K rows × 2K columns
1. A Row in a bank is selected and read using \{BA1,BA0,A13–A0\}
2. Data is moved to I/O latch (row buffer)
3. Part of data is selected using column address
4. Selected data is moved to chip I/O

**DRAM Organization**

- A DIMM consists of 1 – 4 ranks mounted on a single printed-circuit board.
- A Rank consists of multiple (parallel) chips contributing to the same transaction. For example, each of 8 chips can provide a byte for a total of 64 data bits on the channel.
- A memory chip is organized internally as a number of banks (1-8 usually).
- Multiple banks can execute different commands at the same time
Disk storage

- To access data:
  - wait time: until disk is not used for other transactions.
  - seek: position head over the proper track (each surface has a head)
  - rotational latency: wait for desired sector
  - transfer: grab the data (one or more sectors)
  - controller time (overhead)

Example:
- 10,000 – 50,000 track per surface
- 100 - 500 sectors per track,
- 512 Bytes per sector,
- Sync. gaps between sectors
- The number of sectors/track
  → can be fixed, or
  → outer tracks have more sectors than inner tracks

Flash Storage

- Non-volatile semiconductor storage
  - 100X – 1000X faster than disk
  - Smaller, lower power, more robust
  - But more $/GB (between disk and DRAM)

- Flash bits wears out after 1000’s of accesses
  - Not suitable for direct RAM
  - Wear levelling: remap data to less used blocks

NOR flash:
- Lower density
- Random access
- More reliable
- Slower erase
- Faster random read
- Used for instruction memory

NAND flash:
- Higher density
- Page access
- Less reliable (needs ECC)
- Faster erase
- Faster streaming read
- Used for streamed data
Block erasure

- Blocks of cells are organized as pages (rows)
- A page has to be erased before it is written
- A whole block is erased at once
- Cannot erase only part of a block

Flash Translation Layer (FTL):
- Maps virtual pages to physical pages
- Remaps a page on a write
  - invalidate the corresponding page
- Erase blocks with many obsolete pages
  - consolidate valid pages into new blocks

Hybrid hard drive (HHD)

Basic idea
- All discs have a DRAM cache
- HHD adds a Non Volatile (NV) cache using NAND flash memory
Solid-state disk (SSD)

- Instead of magnetic media accessed via mechanical parts (spindle motor, arm, …), SSD uses only solid-state to store information

- NAND flash memories are optimized for storage applications (larger read/write data unit than other addressable memories like SRAM and DRAM)
  - NOR flash has been favored to store codes

- NAND flash memories are susceptible to write endurance
  - Various wear-leveling techniques have been developed
  - Hidden in flash translation layer or FTL

- A fun video: [http://www.youtube.com/watch?v=96dWOEa4Djs](http://www.youtube.com/watch?v=96dWOEa4Djs)