Agenda

- Caches
- SRAM Decoders
- Other Memories (DRAM, CAM, Flash)
Caches
Memory Hierarchy Overview

(B) Memory hierarchy for a laptop or a desktop

What is a Cache?

- A cache holds **commonly used memory data**.

- An **ideal cache** would anticipate all of the data needed by CPU, and fetch it from main memory ahead of time, so that it has zero miss rate.

- Caches are specified by:
  - C: capacity
  - b: block size: Granularity of memory loaded into cache
  - B: number of blocks (B = C / b)
  - S: number of sets
  - N: degree of associativity
Direct Mapped Cache

$N = 1$

- Block address
- Cache Tag: Example: 0x50
- Cache Index
- Offset: Ex: 0x01, Ex: 0x00

Valid Bit

Cache Tag: 0x50

Cache Data
- Byte 31
- Byte 1
- Byte 0
- Byte 63
- Byte 33
- Byte 32
- Byte 1023
- Byte 992

32 blocks per set

block
### Fully Associative Cache

$$N = 32$$

<table>
<thead>
<tr>
<th></th>
<th>Cache Tag (27 bits long)</th>
<th>Byte Select</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>31</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>4</strong></td>
<td></td>
<td>Ex: 0x01</td>
</tr>
<tr>
<td><strong>0</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

32 comparators

#### Compare

- **Cache Tag**
- **Valid Bit**
- **Cache Data**

<table>
<thead>
<tr>
<th></th>
<th>Byte 31</th>
<th>.</th>
<th>Byte 1</th>
<th>Byte 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 63</td>
<td>.</td>
<td>Byte 33</td>
<td>Byte 32</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
<td>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
<td>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
<td>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
N-Way Set Associative Cache

$1 < N < 32$

$N=2$
SRAM Decoders
SRAM Structure:
SRAM Structure:

Address

$N$ words

$2^N$ bits

only 1 will be "1" at a time
Row Decoder: Naive Implementation
Predecoder + Decoder
Other Memories
Content Addressable Memory (CAM)

Match

Mismatch

\[ 1 \oplus 1 = 0 \]
\[ 1 \oplus 0 = 1 \]
DRAM

\[ V_{\text{BL}} = 0 \text{ or } (V_{DD} - V_T) \]
Flash Overview

Flash cell

Charge storage

Floating gate

MOSFET
Flash Overview

Storing 0
OFF State

Storing 1
ON State

Charge storage

Id

Vt_on  Vt_off

V_read

Vgs

1  0
Flash Write

• Step 1: Erasing.
  • Erase all the FG transistors to set them to 1
  • Apply a negative voltage to the gate -> Electrons flow from the floating gate to the substrate.

• Step 2: Programming
  • Reprogram the appropriate FG transistors to set them to 0
  • Apply a high voltage to the gate -> Electrons are tunneled onto the floating gate.
NAND vs NOR Flash

**NAND**
- High Density
- Used for data storage
  - USB drives
  - Memory cards
  - SSD

**NOR**
- Lower Latency
- Used for code storage
  - Embedded systems
NAND Flash Read
NOR Flash Read
Questions?