Cache Memories

Cache memories are small, fast SRAM-based memories managed automatically in hardware.

- Hold frequently accessed blocks of main memory
- CPU looks first for data in L1, then in L2, then in main memory.

Typical bus structure:

```
+-----------------+      +-----------------+
| CPU chip        |      | main memory     |
+-----------------+      +-----------------+
| cache bus       |      | I/O bridge      |
+-----------------+      +-----------------+
| L2 cache        |      | bus interface   |
+-----------------+      +-----------------+
| register file   |      | ALU             |
+-----------------+      +-----------------+
| ALU             |      | system bus      |
+-----------------+      +-----------------+
| memory bus      |      | V0 bridge       |
+-----------------+      +-----------------+
```

Inserting an L1 Cache Between the CPU and Main Memory

The transfer unit between the CPU register file and the cache is a 4-byte block.

The tiny, very fast CPU register file has room for four 4-byte words.

The small fast L1 cache has room for two 4-word blocks.

The big slow main memory has room for many 4-word blocks.

General Org of a Cache Memory

Cache is an array of sets.
Each set contains one or more lines.
Each line holds a block of data.

```
  valid  tag  0  1  ...  d-1

set 0:  ...                        E lines per set

set 1:  ...                        E lines per set

set d-1:  ...                      E lines per set
```

Cache size: $C = B \times E \times S$ data bytes
Direct-Mapped Cache

Simplest kind of cache
Characterized by exactly one line per set.

Accessing Direct-Mapped Caches

Set selection
- Use the set index bits to determine the set of interest.

Accessing Direct-Mapped Caches

Line matching and word selection
- Line matching: Find a valid line in the selected set with a matching tag
- Word selection: Then extract the word

Practical Exercise 1 - Direct-Mapped

M = 64 byte addresses
B = 1 bytes/block
S = 16 sets
E = 1 entry/set
- Address trace (reads):
  1 [000 001], 4 [000 100],
  8 [000 000], 5 [000 101],
  20 [010 000], 17 [010 001],
  19 [010 011], 56 [111 000],
  9 [001 001], 11 [001 011],
  4 [000 100], 43 [101 011],
  5 [000 101], 6 [000 110],
  9 [001 001], 17 [010 001]
- Assuming a direct-mapped cache
  initially empty
- Label each reference in the list as a hit
  or a miss
- Show the final content of the cache
Practical Exercise 2 - Direct-Mapped

M = 64 byte addresses,
B = 4 bytes/block,
S = 4 sets (direct assoc.)
E = 1 entry/set
- Address trace (reads):
  1 [000 001]
  4 [000 100]
  8 [000 101]
  16 [010 000]
  19 [010 011]
  20 [010 100]
  56 [111 000]
  9 [001 001]
  11 [001 011]
  43 [101 011]
  6 [000 110]
  17 [010 001]
  56 [111 000]

- Assume a direct-mapped cache initially empty
- Label each reference in the list as a hit or a miss
- Show the final contents of the cache

M = 64 byte addresses,
B = 4 bytes/block,
S = 4 sets (direct assoc.)
E = 1 entry/set

Set Associative Caches
Characterized by more than one line per set

set 0

set 1

set S-1

Accessing Set Associative Caches

Set selection
- Identical to direct-mapped cache
- Selected set
- Set index
- Tag
- Block offset

Accessing Set Associative Caches
Line matching and word selection
- Must compare the tag in each valid line in the selected set.
- If (1) and (2), then cache hit, and block offset selects starting byte.
### Practical Exercise 3 - 2-way Set Assoc.

- **M** = 64 byte addresses
- **B** = 1 byte/block
- **S** = 8 sets
- **E** = 2 entry/set

Address trace (reads):
- 1 [000 001], 4 [000 100], 8 [001 000], 2 [010 100], 19 [010 011], 5 [000 101], 6 [000 110], 9 [001 001], 11 [001 011], 43 [101 011], 56 [111 000], 17 [010 001], 17 [010 001], 11 [001 011]

• Assume a direct-mapped cache initially empty
• Assume LRU replacement
• Label each reference in the list as a hit or a miss
• Show the final contents of the cache

<table>
<thead>
<tr>
<th>Valid</th>
<th>Tag</th>
<th>Memory</th>
<th>bit</th>
<th>block</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Practical Exercise 4 - Fully Assoc.

- **M** = 64 byte addresses
- **B** = 1 byte/block
- **S** = 1 sets
- **E** = 16 entry/set

Address trace (reads):
- 1 [000 001], 4 [000 100], 8 [001 000], 2 [010 100], 19 [010 011], 5 [000 101], 6 [000 110], 9 [001 001], 11 [001 011], 43 [101 011], 56 [111 000], 17 [010 001], 17 [010 001], 11 [001 011]

• Assume a direct-mapped cache initially empty
• Assume LRU replacement
• Label each reference in the list as a hit or a miss
• Show the final contents of the cache

<table>
<thead>
<tr>
<th>Valid</th>
<th>Tag</th>
<th>Memory</th>
<th>bit</th>
<th>block</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Practical Exercise 5 - Fully Assoc.

- **M** = 64 byte addresses
- **B** = 4 bytes/block
- **S** = 1 sets
- **E** = 4 entry/set

Address trace (reads):
- 1 [000 001], 4 [000 100], 8 [001 000], 2 [010 100], 19 [010 011], 5 [000 101], 6 [000 110], 9 [001 001], 11 [001 011]

• Assume a direct-mapped cache initially empty
• Assume LRU replacement
• Label each reference in the list as a hit or a miss
• Show the final contents of the cache

<table>
<thead>
<tr>
<th>Valid</th>
<th>Tag</th>
<th>Memory</th>
<th>bit</th>
<th>block</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Page <<#>>