Many of the following slides are taken with permission from

Complete Powerpoint Lecture Notes for Computer Systems: A Programmer's Perspective (CS:APP)

Randal E. Bryant and David R. O'Hallaron

http://csapp.cs.cmu.edu/public/lectures.html

The book is used explicitly in CS 2505 and CS 3214 and as a reference in CS 2506.
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 caches (e.g., L1, L2, and L3), then in main memory.

Typical system structure:
General Cache Organization (S, E, B)

**Cache size:**

\( C = S \times E \times B \) data bytes
The "geometry" of the cache is defined by:

\[ S = 2^s \quad \text{the number of sets in the cache} \]
\[ E = 2^e \quad \text{the number of lines (blocks) in a set} \]
\[ B = 2^b \quad \text{the number of bytes in a line (block)} \]

These values define a related way to think about the organization of DRAM:

- DRAM consists of a sequence of blocks of \( B \) bytes.
- The bytes in a block (line) can be indexed by using \( b \) bits.

- DRAM consists of a sequence of groups of \( S \) blocks (lines).
- The blocks (lines) in a group can be indexed by using \( s \) bits.

- Each group contains \( S \times B \) bytes, which can be indexed by using \( s + b \) bits.
Cache (8, 2, 4) and 256-Byte DRAM

Cache size:
\[ C = S \times E \times B = 64 \text{ data bytes} \]

\[ S = 2^3 \text{ sets} \]

\[ E = 2^1 \text{ blocks (lines) per set} \]

\[ B = 2^2 \text{ bytes per cache block (the data)} \]

valid bit

DRAM

0
1
2
3
4
5
6
7
\ldots
252
253
254
255
Assume a cache has the following geometry:

- \( S = 2^2 = 8 \) the number of sets in the cache
- \( E = 2^1 = 2 \) the number of lines (blocks) in a set
- \( B = 2^2 = 4 \) the number of bytes in a line (block)

Suppose that DRAM consists of 256 bytes, so we have 8-bit addresses.

Then DRAM consists of:
- 64 blocks, each holding 4 bytes
- 8 groups, each holding 8 blocks
Example of Cache View of DRAM

Pick an address: 01110101

011 101 01
group  block  byte

\(a_1 a_0\) give the byte number

and the offset of the byte in the block.
Example of Cache View of DRAM

Pick an address: 01110101

011 101 01

group  block  byte

\(a_4a_3a_2\) give the block number

\(a_4a_3a_200\) equals the offset of the block in the group.

\[ * a_4a_3a_200 = a_4a_3a_2 \times 2^2 \]

\[ = a_4a_3a_2 \times \text{(size of a block)} \]
Example of Cache View of DRAM

Pick an address: 01110101

011 101 01
group  block  byte

\( \text{a}_7 \text{a}_6 \text{a}_5 \) give the group number

\( \text{a}_7 \text{a}_6 \text{a}_5 \text{00000} \) equals the offset of the group in the DRAM.

* Why?

* Why?
The BIG Picture

011 101 01

DRAM

00000000
00100000
01000000
01100000
10000000
10100000
11000000
11100000

Group 011 in DRAM

01100000
01100100
01101000
01101100
01110000
01110100
01111000
01111100

Block 101 in Group 011

01110100
01110101
01110110
01110111

byte 00
byte 01
byte 10
byte 11
Example of Cache View of DRAM

Pick an address: 01110101

How does this address map into the cache?

The DRAM block number determines the cache set used to store the block.

Note this means that two DRAM blocks from the same DRAM group cannot map into the same cache set.

So the address: 01110101 maps to set 101 in the cache.

Each set in our cache can hold 2 blocks.

This block could be stored at either location within the corresponding set.
Example of Cache View of DRAM

DRAM block containing address: 01110101

Maps to cache set: 011 101 01

Cache

0 1 2 3

valid tag

Cache Organization 12
So, to generalize, suppose a cache has:

- \( S = 2^s \) sets
- \( E = 2^e \) blocks (lines) per set
- \( B = 2^b \) bytes per block (line)

And, suppose that DRAM uses \( N \)-bit addresses. Then for any address:

\[
\begin{align*}
    a_{N-1} & \ldots a_{s+b} \\
    a_{s+b-1} & \ldots a_b \\
    a_{b-1} & \ldots a_0
\end{align*}
\]

- Bits \( a_{b-1} : a_0 \) give the byte index within the block
- Bits \( a_{b+s-1} : a_b \) give the set index
- Bits \( a_{N-1} : a_{s+b} \) become the tag for the data

Note that these bits are only the same for blocks that are within the same DRAM group.
1. Locate set
2. Check if any line in set has matching tag
3. Yes + line valid: hit
4. Locate data starting at offset
Example: Direct Mapped Cache (E = 1)

Direct mapped: One line per set
Assume: cache block size 8 bytes

S = 2^s sets

Address of int:

\[
\begin{array}{c}
\text{t bits} \\
0...01 \\
100
\end{array}
\]

find set
Example: Direct Mapped Cache (E = 1)

Direct mapped: One line per set
Assume: cache block size 8 bytes

Address of int:

| t bits | 0...01 | 100 |

valid? + matching tags → hit

block offset

v tag 0 1 2 3 4 5 6 7
Example: Direct Mapped Cache (E = 1)

Direct mapped: One line per set
Assume: cache block size 8 bytes

No match: old line (block) is evicted and replaced by requested block from DRAM
Cache Organization

Direct-Mapped Cache Simulation

M=16 byte addresses, B=2 bytes/block, S=4 sets, E=1 Blocks/set

Address trace (reads, one byte per read):

<table>
<thead>
<tr>
<th>Address</th>
<th>Block</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[0000₂]</td>
<td>miss</td>
</tr>
<tr>
<td>1</td>
<td>[0001₂]</td>
<td>hit</td>
</tr>
<tr>
<td>7</td>
<td>[0111₂]</td>
<td>miss</td>
</tr>
<tr>
<td>8</td>
<td>[1000₂]</td>
<td>miss</td>
</tr>
<tr>
<td>0</td>
<td>[0000₂]</td>
<td>miss</td>
</tr>
</tbody>
</table>

v Tag Block

<table>
<thead>
<tr>
<th>Set 0</th>
<th>1</th>
<th>0</th>
<th>M[0-1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set 3</td>
<td>1</td>
<td>0</td>
<td>M[6-7]</td>
</tr>
</tbody>
</table>
E-way Set Associative Cache (Here: E = 2)

E = 2: Two lines per set
Assume: cache block size 8 bytes

Address of short int:
- t bits: 0...01 100

Find set
E-way Set Associative Cache (Here: E = 2)

E = 2: Two lines per set
Assume: cache block size 8 bytes

Address of short int:

\[ t \text{ bits } 0...1 \quad 100 \]

valid? + matching tags \rightarrow hit

compare both

block offset
E-way Set Associative Cache (Here: $E = 2$)

$E = 2$: Two lines per set
Assume: cache block size 8 bytes

Address of short int:

$$t \text{ bits} \quad 0...01 \quad 100$$

compare both

valid? + matching tags $\rightarrow$ hit

block offset

short int (2 Bytes) is here

No match:
- One line in set is selected for eviction and replacement
- Replacement policies: random, least recently used (LRU), ...
2-Way Set Associative Cache Simulation

M=16 byte addresses, B=2 bytes/block, S=2 sets, E=2 blocks/set

Address trace (reads, one byte per read):
0 \[0000_2\], miss
1 \[0001_2\], hit
7 \[0111_2\], miss
8 \[1000_2\], miss
0 \[0000_2\], hit

<table>
<thead>
<tr>
<th>Set 0</th>
<th>Tag</th>
<th>Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00</td>
<td>M[0-1]</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>M[8-9]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Set 1</th>
<th>Tag</th>
<th>Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01</td>
<td>M[6-7]</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The "geometry" of the cache is defined by:

\[
S = 2^s \quad \text{the number of sets in the cache}
\]
\[
E = 2^e \quad \text{the number of lines (blocks) in a set}
\]
\[
B = 2^b \quad \text{the number of bytes in a line (block)}
\]

\[E = 1 \quad (e = 0)\]  
*direct-mapped cache*
only one possible location in cache for each DRAM block

\[S > 1\]
\[E = K > 1\]  
*K-way associative cache*
K possible locations (in same cache set) for each DRAM block

\[S = 1 \quad (\text{only one set})\]
\[E = \# \text{ of cache blocks}\]  
*fully-associative cache*
each DRAM block can be at any location in the cache
Searching a Set

If we have an associative cache (K-way or fully), how do we determine if a given DRAM block occurs within a set?

Compare the tag we’re trying to match to all of the tags for blocks in the relevant set at the same time!

Then factor in the valid bits, also in parallel.

And employ a MUX