ECE6170 Memory Survey
Cache Part III. An Example

Zhe Chen
11/12/2008
Recall ... 

- Increasing cache size reduces miss rate
- Increasing set-associativity reduces miss rate
But ...

- In reality, we have to make a compromise between performance and cost.
- In early real-time embedded systems, there were rarely caches.
- Today it is different. Let’s take a DSP as an example.
DSP

- Digital Signal Processor
  - Focus on data processing
  - Usually used in embedded systems, together with other kinds of processors, like GPPs / MCUs
- TI C64x+ DSP
  - One of the most powerful DSP families
C64x+ Cache Architecture

- 2-level cache
  - L1: separate Program and Data caches
  - L2: unified cache
- Configurable
  - SRAM or Cache
  - Cache capacity
L1 Program Cache

- Organization: direct-mapped
- Line size: 32 bytes
- Protocol: read allocate
- CPU access time: 1 cycle
- Capacity: 4K, 8K, 16K, 32K bytes configurable
L1 Data Cache

- Organization: 2-way set-associative
- Line size: 64 bytes
- Protocol: read allocate, write-back
- CPU access time: 1 cycle
- Capacity: 4K, 8K, 16K, 32K bytes configurable
- Replacement strategy: least recently used (LRU)
L2 Cache

- L2 cache basically operates in the same manner as L1 cache
- Organization: 4-way set-associative
- Line size: 128 bytes
- Protocol: read and write allocate, write-back
- Capacity: 32K, 64K, 128K, 256K bytes configurable
- Replacement strategy: least recently used (LRU)
Think ...

- How to make use of cache to achieve high cache efficiency (i.e., to increase cache hit rate) in programming?
Video scaling is converting video signals from one size or resolution to another: usually "upscaling" a video signal from a lower resolution to one of higher resolutions.
Major Operations in Video Scaling

- Pixel interpolation
  - FIR filtering at pixel level
  - Filtering horizontally or vertically or both
  - Filtering for each channel (Y / Cr / Cb) in each video frame
- The following figure shows an example of 2-times upscaling.

YCbCr 4:2:2 co-sited source pixels

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
</tr>
</thead>
</table>

2x upscaled output

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>b'</th>
<th>c</th>
<th>c'</th>
<th>d</th>
<th>d'</th>
<th>e</th>
<th>e'</th>
<th>f</th>
<th>f'</th>
<th>g</th>
</tr>
</thead>
</table>

Y' = Y

Cb' = Cb

Cr' = Cr

Luma (Y) sample

Chroma (Cb/Cr) samples

Y' = \((-1Y_d + 17Y_c + 17Y_e - 1Y_b) / 32\)

Cb' = \((-1Cb_e + 17Cb_c + 17Cb_e - 1Cb_b) / 32\)

Cr' = \((-1Cr_e + 17Cr_c + 17Cr_e - 1Cr_b) / 32\)
Hints for Programming

- **Code and data partitioning**
  - Break up an algorithm into smaller pieces when either the program is too large to fit into the L1P cache, or the data is too large to fit into the L1D cache.

- **Function grouping**
  - Group the algorithms that work on the same data together in memory.
  - For example, place together the functions that work on the Y luma color path in memory, and place together the Y luma data buffers in memory.

- By doing so, cache efficiency of over 90% can be achieved.
A cache architecture example and an application example are given in this part of cache survey.

Compromise between performance and cost when designing a cache architecture.

Use programming techniques to maximize the cache efficiency.


Thank you!