



BROWN

Computer System Design Lecture 17: Cache Design

Prof. R. Iris Bahar
EN164
March 7, 2007

Reading: Appendix C, Sections C.1, C.2



BROWN

Cache Basics

- What information goes into the cache, in addition to the referenced data or instruction?
- How do we know if a data item is in the cache?
- If it is, how do we find it?

EN164
Lecture 16-2



BROWN

Direct Mapped Cache

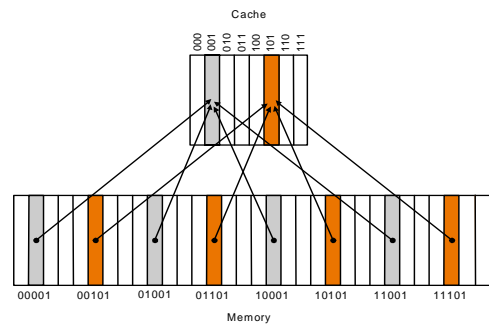
- Simple approach: Direct mapped
 - block size is one word
 - every main memory location can be mapped to exactly one cache location
 - lots of words in the main memory share a single location in the cache
- How is the address composed for the cache?
 - cache address is identical with lower bits in the main memory address
 - tag (higher address bits) differentiates between competing main memory words
- We are taking advantage of **temporal** locality.

EN164
Lecture 16-3



BROWN

Direct Mapped Cache Example



EN164
Lecture 16-4



BROWN

A More Realistic Example

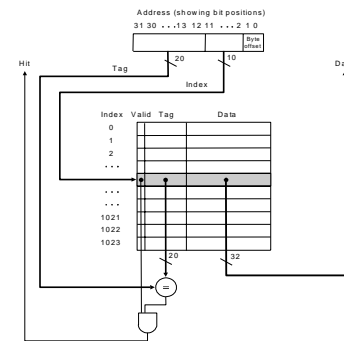
- 32 bit word length
- 32 bit address
- 1 k word cache (=1k X 32bits = 4kB)
- block size 1 word
- 10 bit cache index
 - $2^{10} = 1k$ words
- 2 bit byte offset (word alignment assumed)
- 20 bit tag size
- valid bit

EN164
Lecture 16-5



BROWN

Cache Access



EN164
Lecture 16-6

BROWN

Cache Size

- Cache memory size
 - $1024 \cdot 32 \text{ b} = 32 \text{ kb}$
- Tag memory size
 - $1024 \cdot 20 \text{ b} = 20 \text{ kb}$
- Valid information
 - $1024 \cdot 1 \text{ b} = 1 \text{ kb}$
- Efficiency
 - $32/53 = 60.4\%$

EN164
Lecture 16-7

BROWN

Read Hits and Misses

- Read hits
 - Send desired word back to CPU
 - Cache remains unmodified
- Read misses
 - Stall the CPU
 - Fetch block from next level of memory
 - Update data, tag, valid bit in cache for the requested data
 - Restart the CPU
 - Also requires proper handling of data that was replaced in the cache (to make room for the requested data)

What should happen to this data?

EN164
Lecture 16-9

BROWN

Write Hits and Misses

- Write hits:
 - Update the data in the cache, but what about the next level of memory?
 - **write-through**: every write to L1 \rightarrow write to L2
simplifies coherency protocols in a multiprocessor system but increases memory bus traffic
 - **write-back**: mark the block as *dirty* (i.e., modified), when the block gets replaced from L1, write it to L2
reduces memory bus traffic but complicates coherence protocols
- Write misses:
 - May not be useful to update the data in the cache (writes don't have as high locality). Two options:
 - **Write-no-allocate**: send the write access directly to the next level of memory (leave cache unaltered).
 - **Write-allocate**: read entire block into cache, then write the word
What happens to the replaced word?

EN164
Lecture 16-10

BROWN

Performance

- Simplified model:
 - execution time = (execution cycles + stall cycles) X cycle time
 - stall cycles = # of instructions X miss rate X miss penalty
- The model is more complicated for writes than reads (write-through vs. write-back, write buffers)
- Two ways of improving performance:
 - decreasing the miss rate
 - decreasing the miss penalty

EN164
Lecture 16-12

BROWN

Flexible Placement of Blocks

- Direct mapped cache
 - a memory block can go exactly in one place in the cache
 - use the tag to identify the referenced word
 - easy to implement, but rigid placement can cause high miss rate
- Fully associative cache
 - a memory block can be placed in *any* location in the cache
 - search all entries in the cache in parallel
 - requires a comparator associated with each cache entry
- Set-associative cache
 - a memory block can be placed in a fixed number of locations
 - n locations: n-way set-associative cache
 - a block is mapped to any of n locations in a set
 - Requires searching all locations of the set

EN164
Lecture 16-13