



inst.eecs.berkeley.edu/~cs61c  
**UCB CS61C : Machine Structures**

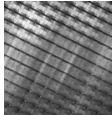
**Lecture 30 – Caches I  
 2008-04-11**

Lecturer SOE  
 Dan Garcia

Hi to Kononov Alexey from Russia !

**WORLD'S FASTEST CHIP: 4.7 GHZ "POWER6"!**

Touted as "the fastest CPU on Earth", IBM's new Power6 doubles the speed of POWER5 but uses the same power. It HAS to be water-cooled, just like the radiators in cars. Their new IBM System p 570 took the top spots in the four most widely used perf benchmarks for Unix servers: SPECint2006, SPECfp2006, SPECjbb2005 and TPC-C, the first time a single system has owned all 4 categories!



tech.slashdot.org/tech/08/04/10/1152205.shtml

**Review : Pipelining**

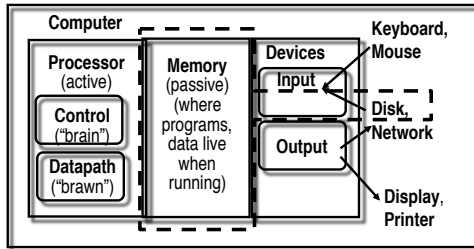
- Pipeline challenge is hazards
  - Forwarding helps w/many data hazards
  - Delayed branch helps with control hazard in our 5 stage pipeline
  - Data hazards w/Loads ⇒ Load Delay Slot
    - Interlock ⇒ "smart" CPU has HW to detect if conflict with inst following load, if so it stalls
- More aggressive performance (discussed in section next week)
  - Superscalar (parallelism)
  - Out-of-order execution



CS61C L30 Caches I (9)

Garcia, Spring 2008 © UCB

**The Big Picture**



CS61C L30 Caches I (8)

Garcia, Spring 2008 © UCB

**Memory Hierarchy**

*I.e., storage in computer systems*

- Processor
  - holds data in register file (~100 Bytes)
  - Registers accessed on nanosecond timescale
- Memory (we'll call "main memory")
  - More capacity than registers (~Gbytes)
  - Access time ~50-100 ns
  - Hundreds of clock cycles per memory access?!
- Disk
  - HUGE capacity (virtually limitless)
  - VERY slow: runs ~milliseconds

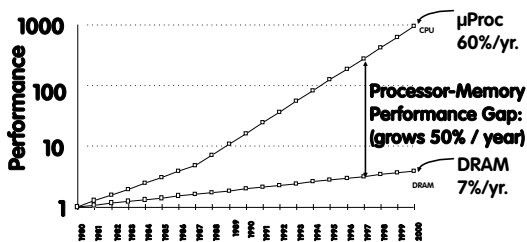


CS61C L30 Caches I (6)

Garcia, Spring 2008 © UCB

**Motivation: Why We Use Caches (written \$)**

- 1989 first Intel CPU with cache on chip
- 1998 Pentium III has two cache levels on chip



CS61C L30 Caches I (5)

Garcia, Spring 2008 © UCB

**Memory Caching**

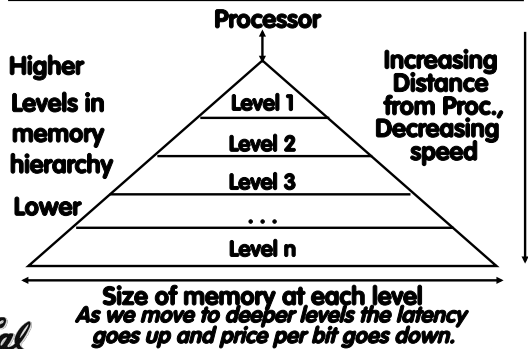
- Mismatch between processor and memory speeds leads us to add a new level: a memory cache
- Implemented with same IC processing technology as the CPU (usually integrated on same chip): faster but more expensive than DRAM memory.
- Cache is a copy of a subset of main memory.
- Most processors have separate caches for instructions and data.



CS61C L30 Caches I (4)

Garcia, Spring 2008 © UCB

## Memory Hierarchy



CSMC L30 Caches | 7 |

©ards, Spring 2008 © UCS

## Memory Hierarchy

- If level closer to Processor, it is:
  - Smaller
  - Faster
  - More expensive
  - subset of lower levels (contains most recently used data)
- Lowest Level (usually disk) contains all available data (does it go beyond the disk?)
- Memory Hierarchy presents the processor with the illusion of a very large & fast memory



CSMC L30 Caches | 8 |

©ards, Spring 2008 © UCS

## Memory Hierarchy Analogy: Library (1/2)

- You're writing a term paper (Processor) at a table in Doe
- Doe Library is equivalent to disk
  - essentially limitless capacity
  - very slow to retrieve a book
- Table is main memory
  - smaller capacity: means you must return book when table fills up
  - easier and faster to find a book there once you've already retrieved it



CSMC L30 Caches | 9 |

©ards, Spring 2008 © UCS

## Memory Hierarchy Analogy: Library (2/2)

- Open books on table are cache
  - smaller capacity: can have very few open books fit on table; again, when table fills up, you must close a book
  - much, much faster to retrieve data
- Illusion created: whole library open on the tabletop
  - Keep as many recently used books open on table as possible since likely to use again
  - Also keep as many books on table as possible, since faster than going to library



CSMC L30 Caches | 10 |

©ards, Spring 2008 © UCS

## Memory Hierarchy Basis

- Cache contains copies of data in memory that are being used.
- Memory contains copies of data on disk that are being used.
- Caches work on the principles of temporal and spatial locality.
  - Temporal Locality: if we use it now, chances are we'll want to use it again soon.
  - Spatial Locality: if we use a piece of memory, chances are we'll use the neighboring pieces soon.



CSMC L30 Caches | 11 |

©ards, Spring 2008 © UCS

## Cache Design

- How do we organize cache?
- Where does each memory address map to?
  - (Remember that cache is subset of memory, so multiple memory addresses map to the same cache location.)
- How do we know which elements are in cache?
- How do we quickly locate them?



CSMC L30 Caches | 12 |

©ards, Spring 2008 © UCS

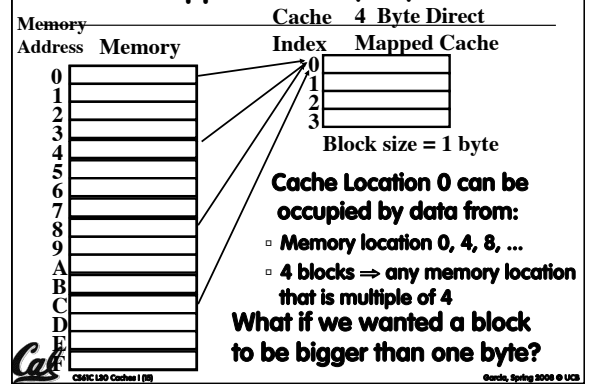


- **Project 4 (on Caches) will be in optional groups of two.**
  - Dan's story of "Partner" Roberto Sanchez-Ramos
- **We're trying to schedule Faux Exam 3**
  - We're hoping to get a room 2008-04-21
  - It'll have different Qs than Faux Exam 2
- **The project is now due friday!**
  - Thanks for your feedback ; we listen!!!
- **Look for the performance competition to be up over the weekend**
  - Start finding partners

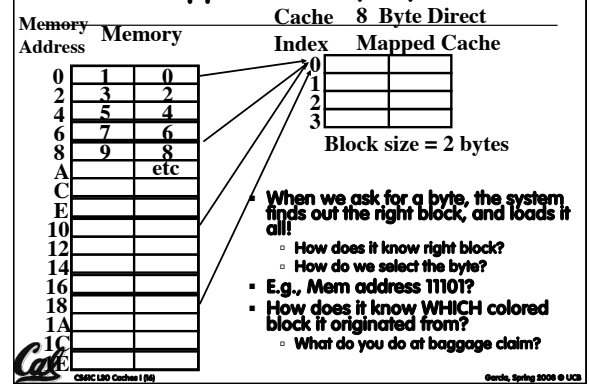
### Direct-Mapped Cache (1/4)

- In a direct-mapped cache, each memory address is associated with one possible block within the cache
  - Therefore, we only need to look in a single location in the cache for the data if it exists in the cache
  - Block is the unit of transfer between cache and memory

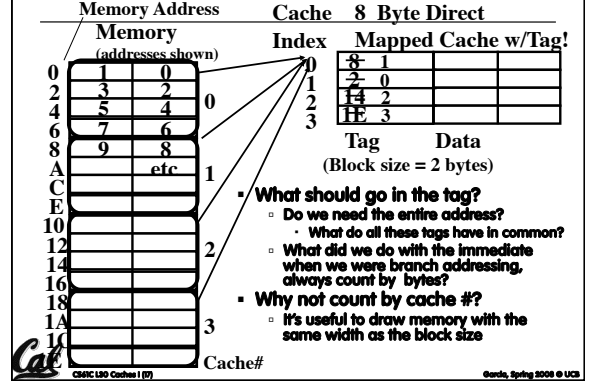
### Direct-Mapped Cache (2/4)



### Direct-Mapped Cache (3/4)

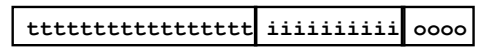


### Direct-Mapped Cache (4/4)



### Issues with Direct-Mapped

- Since multiple memory addresses map to same cache index, how do we tell which one is in there?
- What if we have a block size > 1 byte?
- Answer: divide memory address into three fields



tag to check if have correct block      index to select block      byte offset within block

## Direct-Mapped Cache Terminology

- All fields are read as unsigned integers.
- Index
  - specifies the cache index (which "row"/block of the cache we should look in)
- Offset
  - once we've found correct block, specifies which byte within the block we want
- Tag
  - the remaining bits after offset and index are determined; these are used to distinguish between all the memory addresses that map to the same location

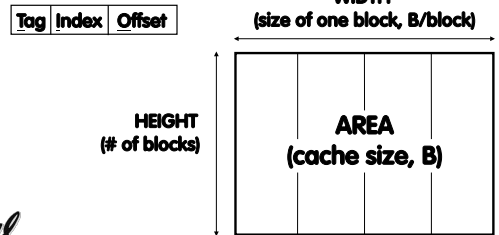
Cal

CS16C L10 Caches | 18

Stanford, Spring 2008 © UCS

## TIO Dan's great cache mnemonic

AREA (cache size, B)  
= HEIGHT (# of blocks)  $2^{H+W} = 2^H * 2^W$   
\* WIDTH (size of one block, B/block)



Cal

CS16C L10 Caches | 18

Stanford, Spring 2008 © UCS

## Direct-Mapped Cache Example (1/3)

- Suppose we have a 8B of data in a direct-mapped cache with 2 byte blocks
  - Sound familiar?
- Determine the size of the tag, index and offset fields if we're using a 32-bit architecture
- Offset
  - need to specify correct byte within a block
  - block contains 2 bytes  
=  $2^1$  bytes
  - need 1 bit to specify correct byte

Cal

CS16C L10 Caches | 18

Stanford, Spring 2008 © UCS

## Direct-Mapped Cache Example (2/3)

- Index: (~index into an "array of blocks")
  - need to specify correct block in cache
  - cache contains 8 B =  $2^3$  bytes
  - block contains 2 B =  $2^1$  bytes
  - # blocks/cache
    - = bytes/cache  
bytes/block
    - =  $2^3$  bytes/cache  
 $2^1$  bytes/block
    - =  $2^2$  blocks/cache
  - need 2 bits to specify this many blocks

Cal

CS16C L10 Caches | 18

Stanford, Spring 2008 © UCS

## Direct-Mapped Cache Example (3/3)

- Tag: use remaining bits as tag
  - tag length = addr length - offset - index  
= 32 - 1 - 2 bits  
= 29 bits
  - so tag is leftmost 29 bits of memory address
- Why not full 32 bit address as tag?
  - All bytes within block need same address (4b)
  - Index must be same for every address within a block, so it's redundant in tag check, thus can leave off to save memory (here 10 bits)

Cal

CS16C L10 Caches | 18

Stanford, Spring 2008 © UCS

## And in Conclusion...

- We would like to have the capacity of disk at the speed of the processor: unfortunately this is not feasible.
- So we create a memory hierarchy:
  - each successively lower level contains "most used" data from next higher level
  - exploits temporal & spatial locality
  - do the common case fast, worry less about the exceptions  
(design principle of MIPS)
- Locality of reference is a Big Idea

Cal

CS16C L10 Caches | 18

Stanford, Spring 2008 © UCS