Computer Science 146
Computer Architecture

Fall 2019

Harvard University

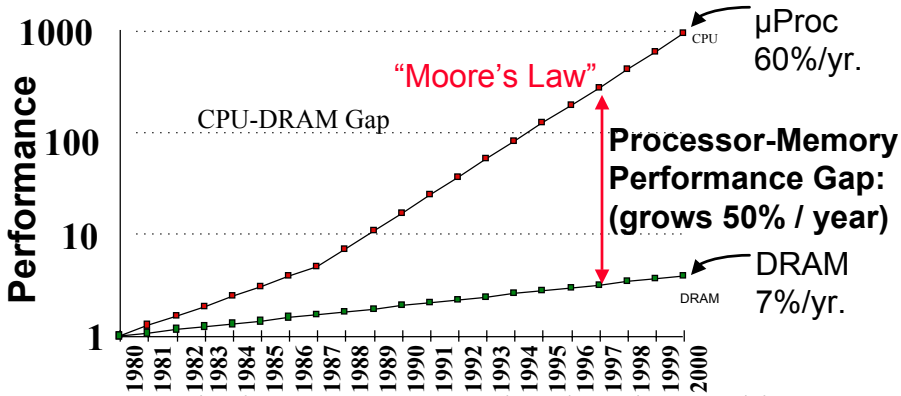
Instructor: Prof. David Brooks
dbrooks@eecs.harvard.edu

Lecture 14: Introduction to Caches

Lecture Outline

- Midterm Review
- Project Discussion (1 Page proposals due 1 week later, April 12th)
- Memory Hierarchy Overview
- Caches

Why care about the Memory Hierarchy?



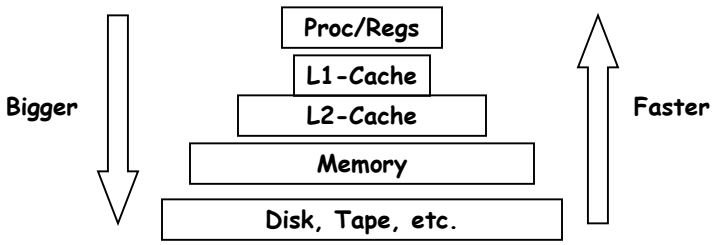
1980: no cache in µproc; 1995 2-level cache on chip
(1989 first Intel µproc with a cache on chip)

Memory Hierarchy Design

- Until now we have assumed a very ideal memory
  - All accesses take 1 cycle
- Assumes an unlimited size, very fast memory
  - Fast memory is very expensive
  - Large amounts of fast memory would be slow!
- Tradeoffs
  - Cost-speed and size-speed
- Solution:
  - Smaller, faster expensive memory close to core "cache"
  - Larger, slower, cheaper memory farther away

What is a cache?

- Small, fast storage used to improve average access time to slow memory
- Hold subset of the instructions and data used by program
- Exploits spacial and temporal locality



Caches are everywhere

- In computer architecture, almost everything is a cache!
  - Registers "a cache" on variables – software managed
  - First-level cache a cache on second-level cache
  - Second-level cache a cache on memory
  - Memory a cache on disk (virtual memory)
  - TLB a cache on page table
  - Branch-prediction a cache on prediction information?

Memory Hierarchy Specs

Type	Capacity	Latency	Bandwidth
Register	<2KB	1ns	150GB/s
L1 Cache	<64KB	4ns	50GB/s
L2 Cache	<8MB	10ns	25GB/s
L3 Cache	<64MB	20ns	10GB/s
Memory	<4GB	50ns	4GB/s
Disk	>1GB	10ms	10MB/s

Program locality is why caches work

- Memory hierarchy exploit program locality:
  - Programs tend to reference parts of their address space that are local in time and space
  - Temporal locality: recently referenced addresses are likely to be referenced again (reuse)
  - Spatial locality: If an address is referenced, nearby addresses are likely to be referenced soon
- Programs that don't exploit locality won't benefit from caches

Locality Example

```
j=val1;
k=val2;
For (i=0; i<10000;i++) {
  A[i] += j;
  B[i] += k;}
```

- Data Locality: i,A,B,j,k?
- Instruction Locality?

Terminology

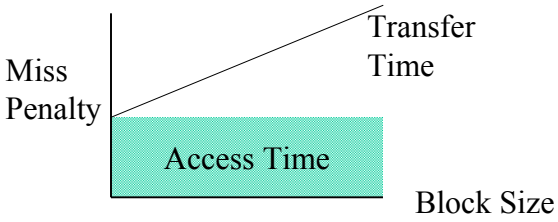
- Higher levels in the hierarchy are closer to the CPU
- At each level a block is the minimum amount of data whose presence is checked at each level
  - Blocks are also often called lines
  - Block size is always a power of 2
  - Block sizes of the lower levels are usually fixed multiples of the block sizes of the higher levels

Terminology (2)

- A reference is said to hit at a particular level if the block is found at that level
  - Hit Rate (HR) = Hits/References
  - Miss Rate (MR) = Misses/References
- Access time of a hit is the hit time
- The additional time to fetch a block on a miss is called the miss penalty

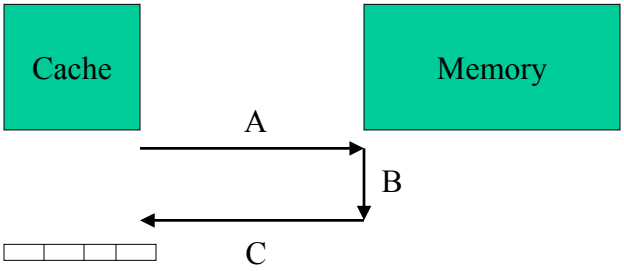
Terminology (3)

- Miss Penalty = Access Time + Transfer Time



- Access time is a function of latency
  - "Time for memory to get request and process it"
- Transfer time is a function of bandwidth
  - "Time for all of block to get back"

Access vs. Transfer Time



- Miss penalty = A + B + C
- Access Time = A + B
- Transfer Time = C

Latency vs. Bandwidth

"There is an old network saying: Bandwidth problems can be cured with money. Latency problems are harder because the speed of light is fixed --- you can't bribe God."

David Clark, MIT

- Recall pipelining: We improved instruction bandwidth, but actually made latency worse
- With memory, for bandwidth we can:
  - Wider buses, larger block sizes, new DRAM organizations (RAMBUS)
- Latency is still much harder:
  - Have to get request from cache to memory (off chip)
  - Have to do memory lookup
  - Have to have bits travel on wire back on-chip to cache

Memory Hierarchy Performance

- Misleading indicator of memory hierarchy performance is miss rate
- Average memory access time (AMAT) is better, but execution time is always the best
- AMAT = Hit Time + Miss Rate * Miss Penalty

Example:

Hit Time = 1ns, Miss Penalty = 20ns, Miss Rate = 0.1

AMAT = 1 + 0.1 * 20 = 3ns

Caching Basics

- Most Basic Caching questions:
  - How do we know if a data item is in the cache?
  - If it is, how do we find it?
  - If it isn't, how do we get it?

More Detailed Questions

- Block placement policy?
  - Where does a block go when it is fetched?
- Block identification policy?
  - How do we find a block in the cache?
- Block replacement policy?
  - When fetching a block into a full cache, how do we decide what other block gets kicked out?
- Write strategy?
  - Does any of this differ for reads vs. writes?

General View of Caches

- Cache is made of frames
  - Frame = data + tag + state bits
  - State bits: Valid (tag/data there), Dirty (wrote into data)
- Cache Algorithm
  - Find frame(s)
  - If incoming tag != stored tag then Miss
    - Evict block currently in frame
    - Replace with block from memory (or L2 cache)
  - Return appropriate word within block

Simple Cache Example

- Direct-mapped cache: Each block has a specific spot in the cache. IF it is in the cache, only one place for it
- Makes block placement, ID, and replacement policies easy
  - Block Placement: Where does a block go when fetched?
    - It goes to its one assigned spot
  - Block ID: How do we find a block in the cache?
    - We look at the tags for that one assigned spot
  - Block replacement: What gets kicked out?
    - Whatever is in its assigned spot
  - Write strategy:
    - "Allocate on write" More on write strategies later

Cache Example continued

- 8 locations in our cache
- Block Size = 1 byte
- Data reference stream:
- References to memory locations:
  - 0,1,2,3,4,5,6,7,8,9,0,0,0,2,2,2,4,9,1,9,1
- Entry = address mod cache size

Lower order bits →

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Cache Examples: Cycles 1 - 5

0,1,2,3,4,5,6,7,8,9,0,0,0,2,2,2,4,9,1,9,1

Miss	Miss	Miss	Miss	Miss
0	0	0	0	0
	1	1	1	1
		2	2	2
			3	3
				4

Cache Examples: Cycles 6-10

0,1,2,3,4,5,6,7,8,9,0,0,0,2,2,2,4,9,1,9,1

Miss	Miss	Miss	Miss	Miss
0	0	0	8	8
1	1	1	1	9
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5
	6	6	6	6
		7	7	7

Cache Examples: Cycles 11-15

0,1,2,3,4,5,6,7,8,9,0,0,0,2,2,2,4,9,1,9,1

Miss	Hit (0)	Hit (0)	Hit (2)	Hit (2)
0	0	0	0	0
9	9	9	9	9
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5
6	6	6	6	6
7	7	7	7	7

Cache Examples: Cycles 16-21

0,1,2,3,4,5,6,7,8,9,0,0,0,2,2,2,4,9,1,9,1

Hit(2)	Hit (4)	Hit(9)	Miss	Miss	Miss
0	0	0	0	0	0
9	9	9	1	9	1
2	2	2	2	2	2
3	3	3	3	3	3
4	4	4	4	4	4
5	5	5	5	5	5
6	6	6	6	6	6
7	7	7	7	7	7

Example Summary

- Hit Rate = 7/21 = 1/3
- Miss Rate = 14/21 = 2/3

- Now, what if the block size = 2 bytes?
- Entry = $\lfloor$Addr/Block Size$\rfloor$ mod Cache Size

Next Time

- Type and Reasons for Cache Misses
- Improving cache performance
- Write policies