ECSE 425 Lecture 19: Memory Hierarchy (an Introduction)

H&P Appendix C

Last Time

- Advanced Techniques
  - Instruction delivery
  - Speculation
- ILP review

Today

- Memory Hierarchy
  - Introduction
  - Caching Basics (Appendix C.1)

Unlimited Amounts of Fast Memory?

"Ideally one would desire an indefinitely large memory capacity such that any particular…word would be immediately available… We are…forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible."

A. W. Burks, H. H. Goldstine, and J. von Neumann, 1946

The Memory Performance "Wall"

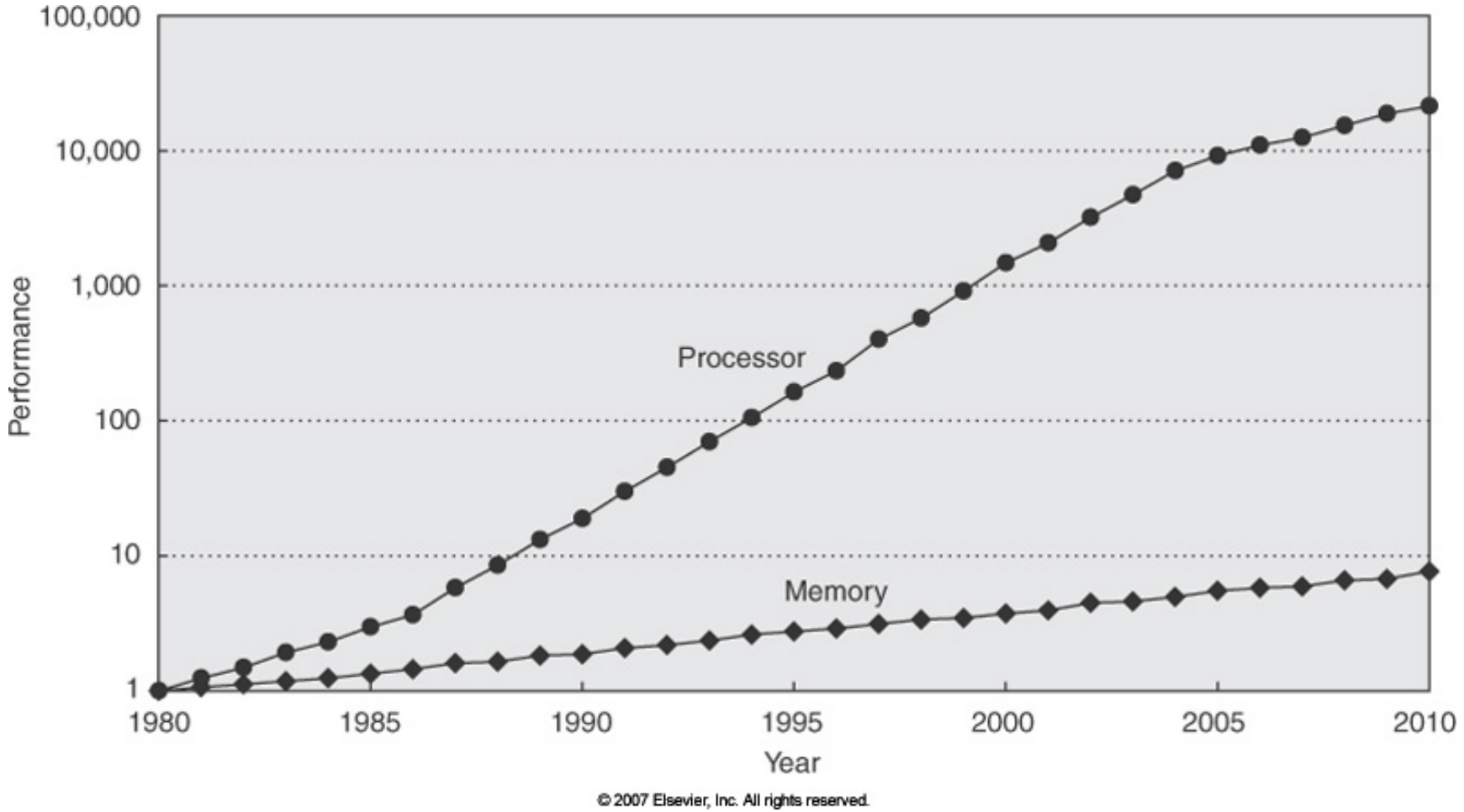


Figure 5.2 Processor-DRAM performance gap. The 1.07 improvement in DRAM latency per year is dominated by improvement in processor performance (1.25 to 1986, 1.52 to 2004, and 1.2 thereafter).

Recall the Principle of Locality

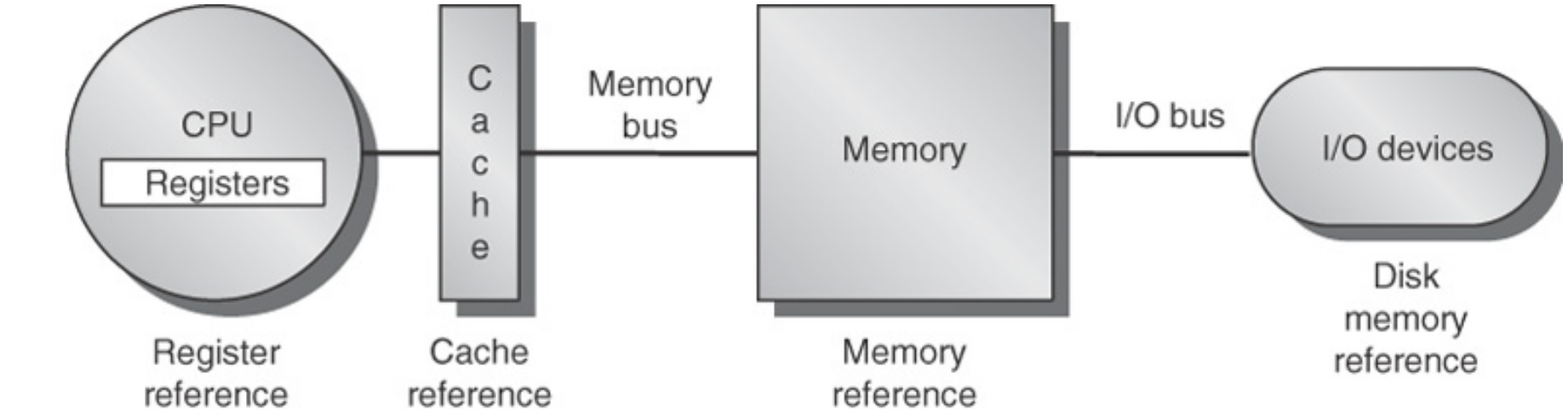
- Any given memory access gives you information about the next likely set of accesses
- Temporal locality
  - Recently accessed items are likely to be accessed again soon—loop, reuse
- Spatial locality
  - Items with nearby addresses tend to be referenced nearby in time—code without branching, arrays

Exploiting Locality with Memory Hierarchy

- Problem: fast memory is expensive!
  - Area per bit, energy per access
- Solution: hierarchy of memory
  - Organized into different levels
  - Each level is progressively larger, but also slower
- It works because of locality
  - Amortize the latency of a few accesses to main memory because cached data is accessed many times

Memory Hierarchy

- Apparently contradictory requirements:
  - Provide virtually unlimited storage
  - Allow the CPU to work at register speed



Typical Memory Hierarchy (circa 2006)

Level	1 registers	2 cache	3 main memory	4 disk storage
Typical size	< 1 KB	< 16 MB	< 512 GB	> 1 TB
Technology	Multi-port custom CMOS memory	On-chip or off-chip CMOS SRAM	CMOS DRAM	Magnetic disk
Access time (ns) (in 2006)	0.25 - 0.5	0.5 - 25	50 - 250	5,000,000
Bandwidth (Mb/s)	50,000 - 500,000	5000 - 20,000	2500 - 10,000	50 - 500
Managed by	compiler	hardware	OS	OS/people
Next level	cache	main memory	disk	CD/tape

What's a cache?

- A small storage array
  - For each entry, data, and identifying information
- Buffers data between registers and main memory
  - Stores a copy of data from main memory
- Multiple caches can be used
  - Small, fast caches close to the CPU
  - Larger, slower caches close to main memory
  - Hides the latency of accessing main memory by limiting the number of times it is accessed

Caching to Reduce Access Delay

- Object: keep in higher level (faster) storage a copy of a subset of the next lower level (slower)
  - Registers hold a subset of cache blocks
  - Cache holds a subset of main memory
  - Main memory holds a subset of virtual memory
- Optimization: keep data likely to be needed soon, minimizing the time to access it
  - Caches are divided into blocks to leverage locality
  - Fast access to previously accessed data
  - Fast access to data we'll likely need in the future

Cache Basics

- CPU requests an item (instruction fetch or load)
  - If it is found in the next lower level, cache hit!
  - If it not found, cache miss
- Cache hit
  - Deliver the item
- Cache miss
  - Attempt to retrieve the data from main memory
  - Not found in main memory? Page fault
    - The page is retrieved from the disk

CPU Performance with Caches

- In-order CPU: CPU stalls on cache miss
- Out-of-order CPU: other instructions may proceed (hiding some latency)
- For now, assume in-order CPU

CPU time = (IC × CPI + Memory stall cycles) × CT

$$\begin{aligned}\text{MemoryStallCycles} &= \text{NumberOfMisses} \times \text{MissPenalty} \\ &= \text{IC} \times \frac{\text{Misses}}{\text{Instructions}} \times \text{MissPenalty} \\ &= \text{IC} \times \frac{\text{MemoryAccesses}}{\text{Instruction}} \times \text{MissRate} \times \text{MissPenalty}\end{aligned}$$

Miss Rate

- What is the miss rate?
  - One of two important metrics in cache design
  - The average number of misses per cache access
  - Different from misses per instruction
- Measure it using a cache simulator
  - Generate an address trace—the sequence of memory addressed accessed—using a performance simulator
  - Cache simulator determines which accesses hit, miss
- Measure it with performance counters
  - Processor reports the number of misses

Separating Read and Write Stalls

- Miss rate and miss penalty are often different for reads and writes

$$\begin{aligned}\text{MemoryStallCycles} &= \text{IC} \times \frac{\text{MemoryAccesses}}{\text{Instruction}} \times \text{MissRate} \times \text{MissPenalty} \\ &= \text{IC} \times \text{ReadsPerInstruction} \times \text{ReadMissRate} \times \text{ReadMissPenalty} \\ &+ \text{IC} \times \text{WritesPerInstruction} \times \text{WritesMissRate} \times \text{WritesMissPenalty}\end{aligned}$$

CPU Performance Example

- Assume
  - CPI = 1 when all cache hits
  - 50% of instructions are loads and stores
  - Miss penalty of 25 cycles
  - Miss rate of 2%
- How much faster would the CPU be if there were no misses?

Four Memory Hierarchy Questions

Main memory is divided into blocks each consisting of several data elements (*e.g.* bytes)

1. Block placement
   – Where can a block be placed in the upper level?
2. Block identification
   – How is a block found if it is in the upper level?
3. Block replacement
   – Which block should be replaced on a miss?
4. Write strategy
   – What happens on a write?

Summary

- Memory hierarchy
  - Creates the illusion of fast, limitless memory
  - Hides latency by using multiple levels of storage
- Close to the CPU, small caches
  - Store a subset of data in memory
  - Caches get progressively larger and slower as the hierarchy approaches main memory and disk

Next Time

- Four Questions (Appendix C.1)
  - Q1: Block placement
  - Q2: Block identification
  - Q3: Block replacement
  - Q4: Write strategy