

High Performance Computing

(CS 680)

Lecture 3: Memory Hierarchy Design*

Jeremy R. Johnson

Wed. Nov. 15, 2000

*This lecture was derived from material in H&P (Chap. 5 sec. 1-5).

Introduction

- **Objective: To design a memory system that provides the illusion of a very large memory that can be accessed as fast as a very small memory.**
- **Principle of Locality Programs tend to access a relatively small portion of their address space at an instant of time.**
- **Topics**
 - memory hierarchy and temporal and spatial locality
 - The basics of cache
 - Cache performance
 - miss rate
 - miss penalty
 - Improving cache performance
 - associativity
 - multi-level caches

Processor-Memory Gap

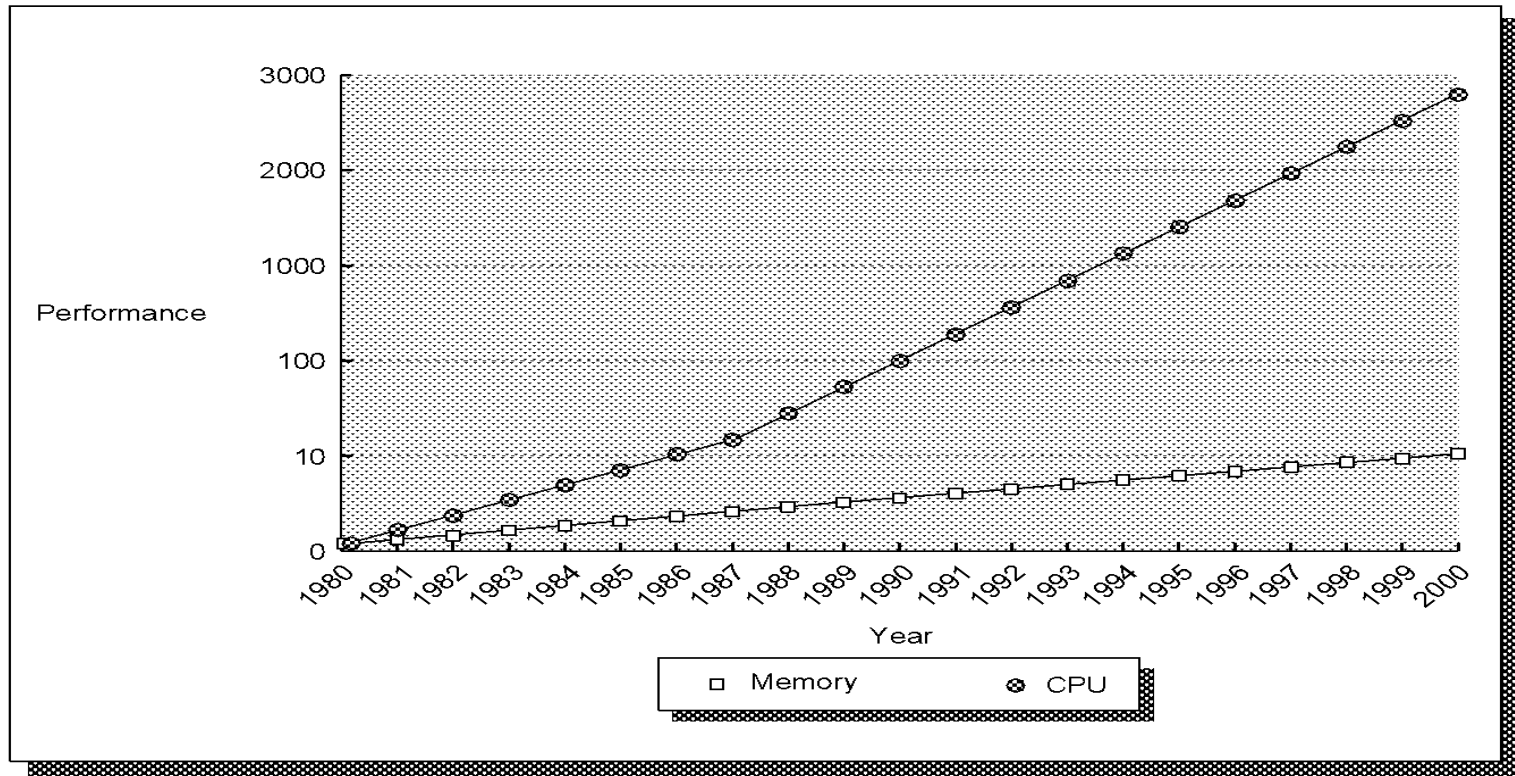
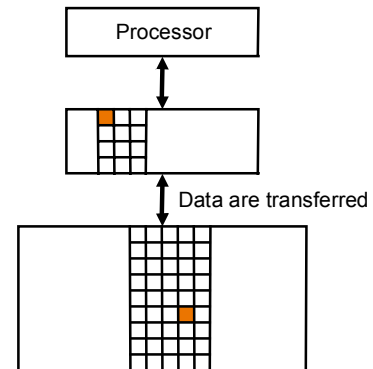
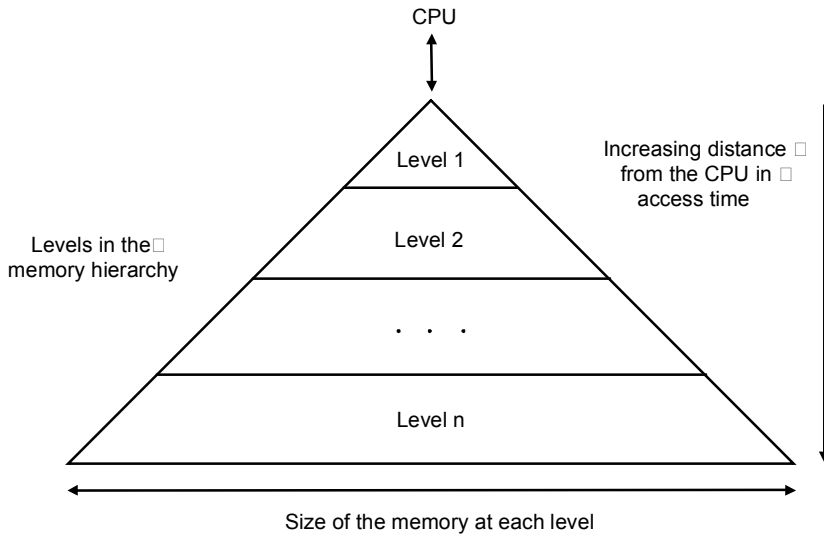


FIGURE 5.1 Starting with 1980 performance as a baseline, the performance of memory and CPUs are plotted over time.

Memory Hierarchy

Memory Technology	Typical Access Time	\$/Megabyte (1997)
SRAM	5-25 ns	\$100-\$250
DRAM	60-120 ns	\$5-\$10
Magnetic Disk	10-20 million ns	\$0.10-\$0.20



Memory Access Speed on DEC 21164 Alpha

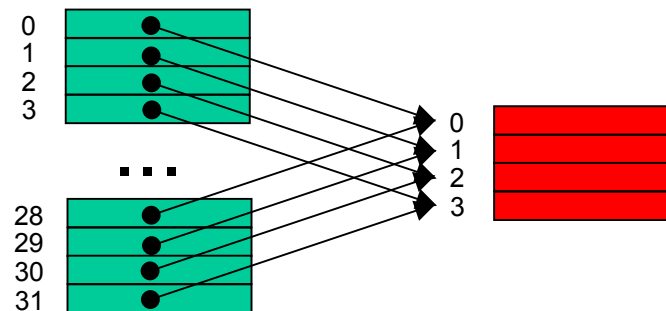
- **Clock Speed 500 MHz (= 2 ns clock rate)**
- **Registers (2 ns)**
- **L1 On-Chip (4 ns)**
- **L2 On-Chip (5 ns)**
- **L3 Off-Chip (30 ns)**
- **Memory (220 ns)**

Common Framework for Memory Hierarchies

- **Question 1: Where can a block be placed?**
 - One place (direct mapped), a few places (set associative), or any place (fully associative)
- **Question 2: How is a block found?**
 - There are four methods: indexing, limited search, full search, or table lookup
- **Question 3: Which block should be replaced on a cache miss?**
 - Typically least recently used or random block
- **Question 4: What happens on writes?**
 - Write-through or write-back

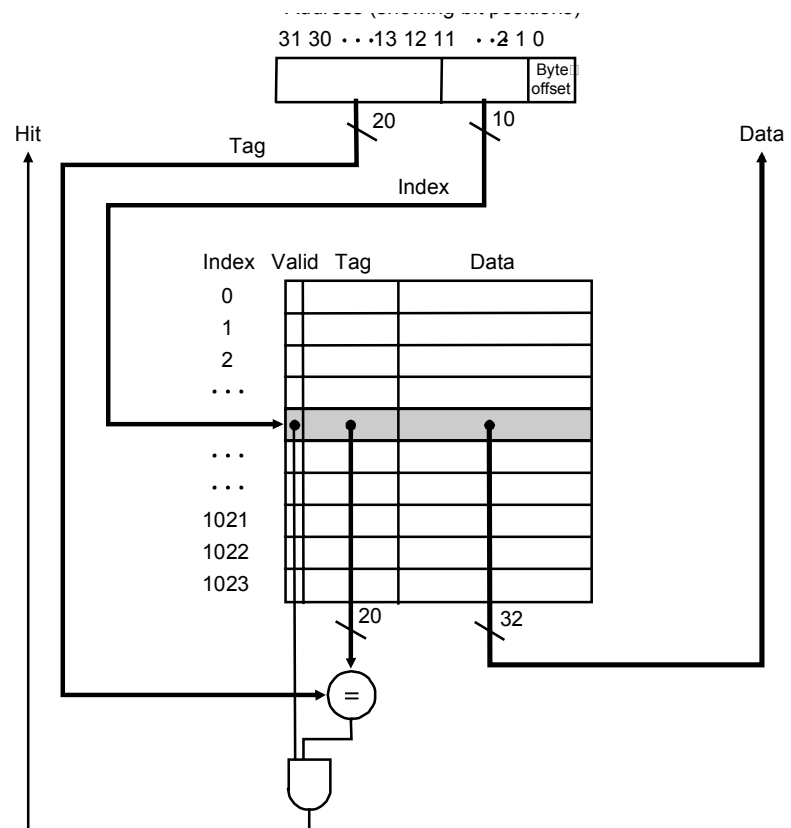
Mapping to Cache

- **Cache** - the level of the memory hierarchy between the CPU and main memory.
- **Direct-Mapped Cache** - memory mapped to one location in cache
 $(\text{Block address}) \bmod (\text{Number of block in cache})$
- **Number of blocks is typically a power of two \Rightarrow cache location obtained from low-order bits of address.**



Locating an Data in the Cache

- Compare cache index (mapping) to Tag (high-order bits) to see if element is currently in cache
- Valid bit used to indicate whether data in cache is valid
- A hit occurs when the data is in cache, otherwise it is a miss
- The extra time required when a cache miss occurs is called the miss penalty



Example

- **32-word memory**
- **8-word cache**

Address	Binary	Cache block	Hit or miss
22	10110	110	
26	11010	010	
22	10110	110	
26	11010	010	
16	10000	000	
3	00011	011	
16	10000	000	
18	10010	010	

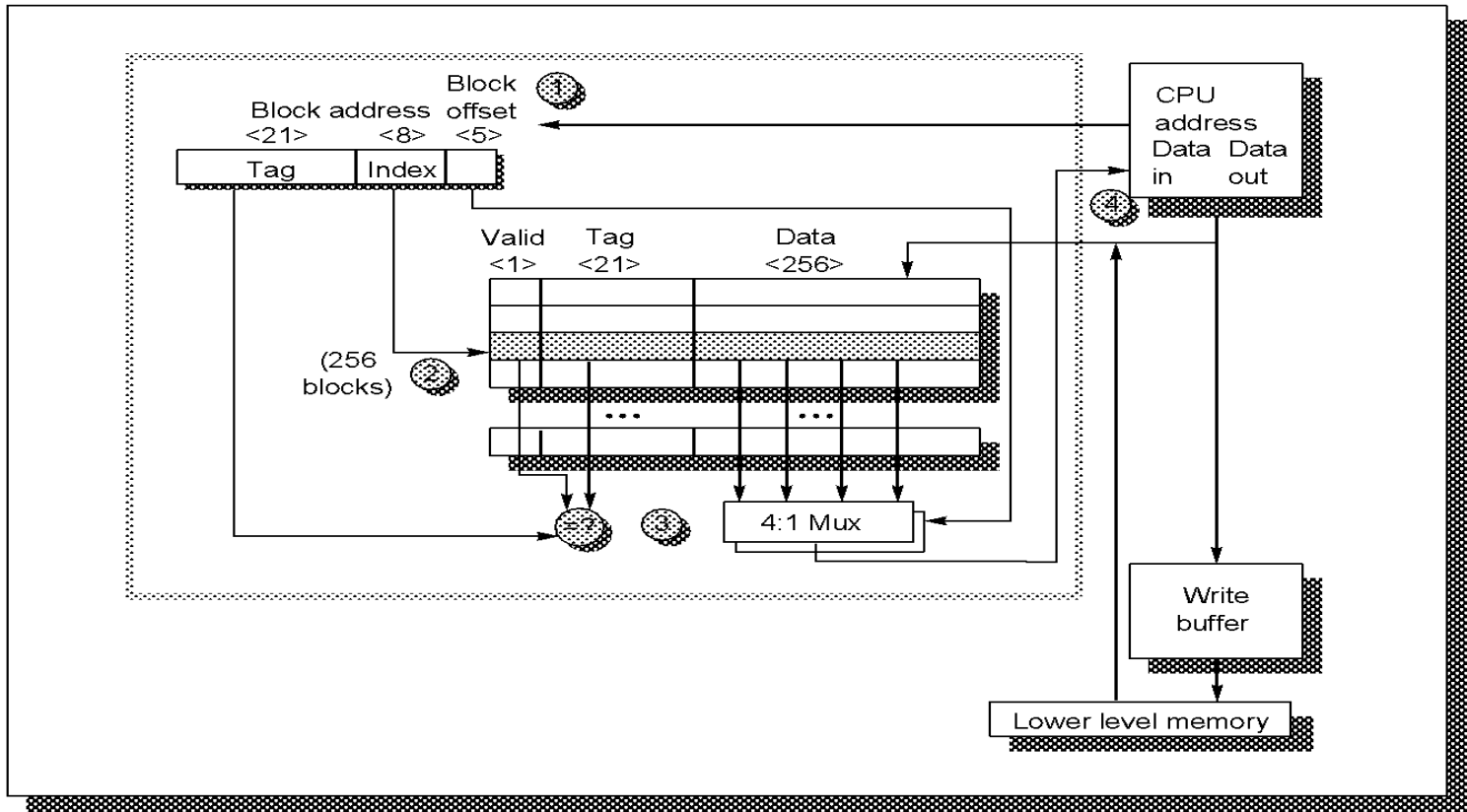
Index	Valid	Tag	Data
000			
001			
010			
011			
100			
101			
110			

Example: Alpha AXP 21064

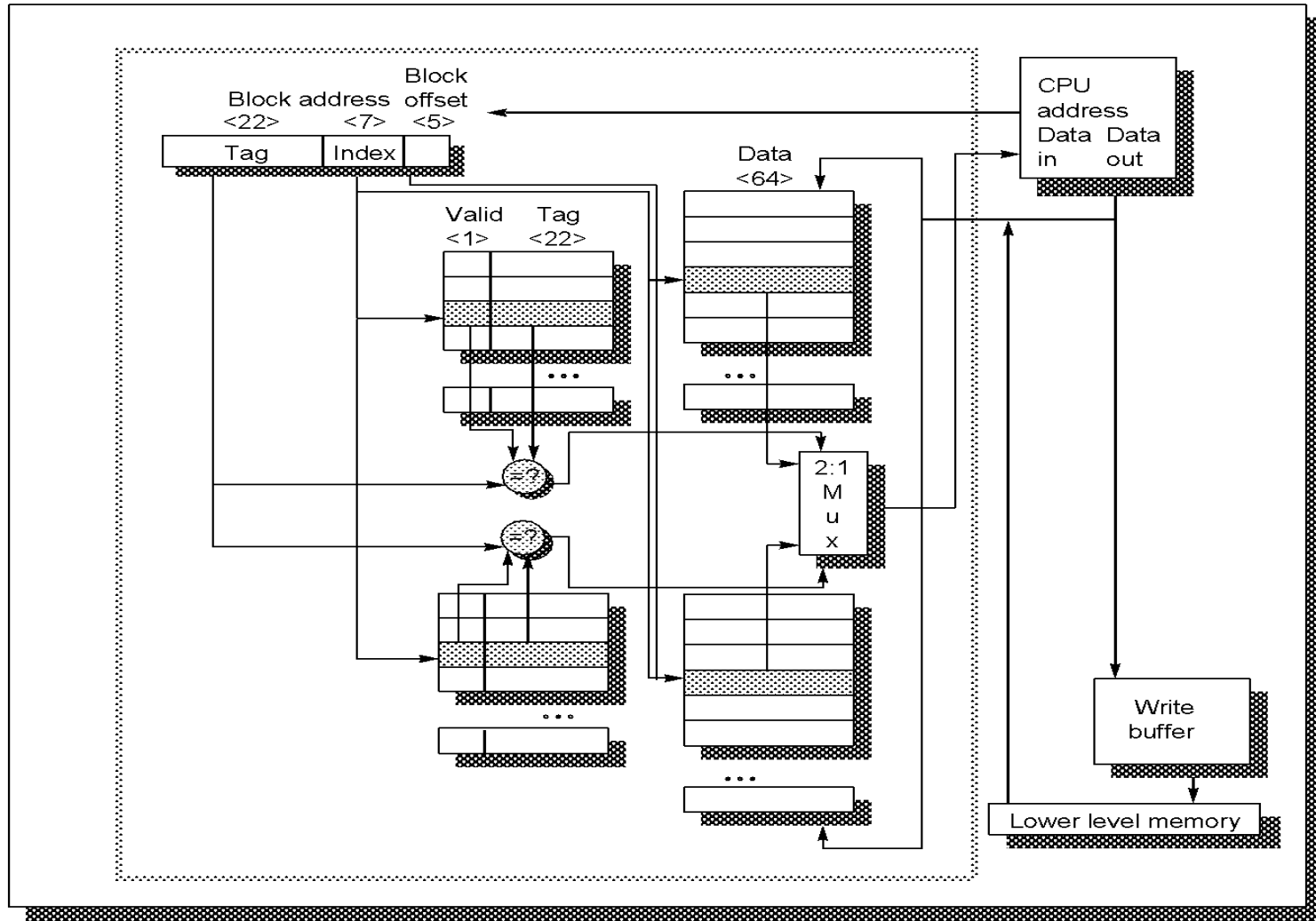
- 8KB cache
- 32-byte blocks
- direct mapped
- write through
- 4-block write buffer
- write around

- Figure 5.5 (page 381)

Alpha AXP 21064



2-Way Set Associative Cache



Cache Organization

- **Since cache is smaller than memory more than one address must map to same line in cache**
- **Direct-Mapped Cache**
 - address mod cache size (only one location when memory address gets mapped to)
- **Fully Associative Cache**
 - address can be mapped anywhere in cache
 - need tag and associative search to find if element in cache
- **Set-Associative Cache**
 - compromise between two extremes
 - element can map to several locations

Model for Cache Misses

- **Compulsory misses**
 - These are cache misses caused by the first access to a block that has never been in the cache
- **Capacity misses (cold-start)**
 - These are cache misses caused when the cache cannot contain all the blocks needed during execution of a program.
- **Conflict misses (collision)**
 - These are cache misses that occur in a set associative or direct mapped cache when multiple blocks compete for the same set. These misses are eliminated with a fully associative cache.

Reducing Miss Rate

- **Larger block size (reduce compulsory misses - may increase conflict and capacity misses)**
 - increase miss penalty
- **Higher associativity (reduce conflict misses)**
 - increase hit time
- **Victim cache**
- **Pseudo-associative cache**
- **Hardware controlled prefetching**
- **Software controlled prefetching**
- **Compiler optimizations**

Reducing Miss Penalty

- **Write buffer**
- **Giving priority to read misses**
- **Subblocks**
- **Early restart and critical word first**
- **Non-blocking caches**
- **Second level caches**

Reducing Hit Time

- **Simple and small caches**
- **Avoiding address translation during indexing of the cache**
- **Pipelining writes for fast write hits**
 - write hits usually take longer than read hits since writing must wait for tag match (to avoid overwriting data)

Taking Advantage of Spatial Locality

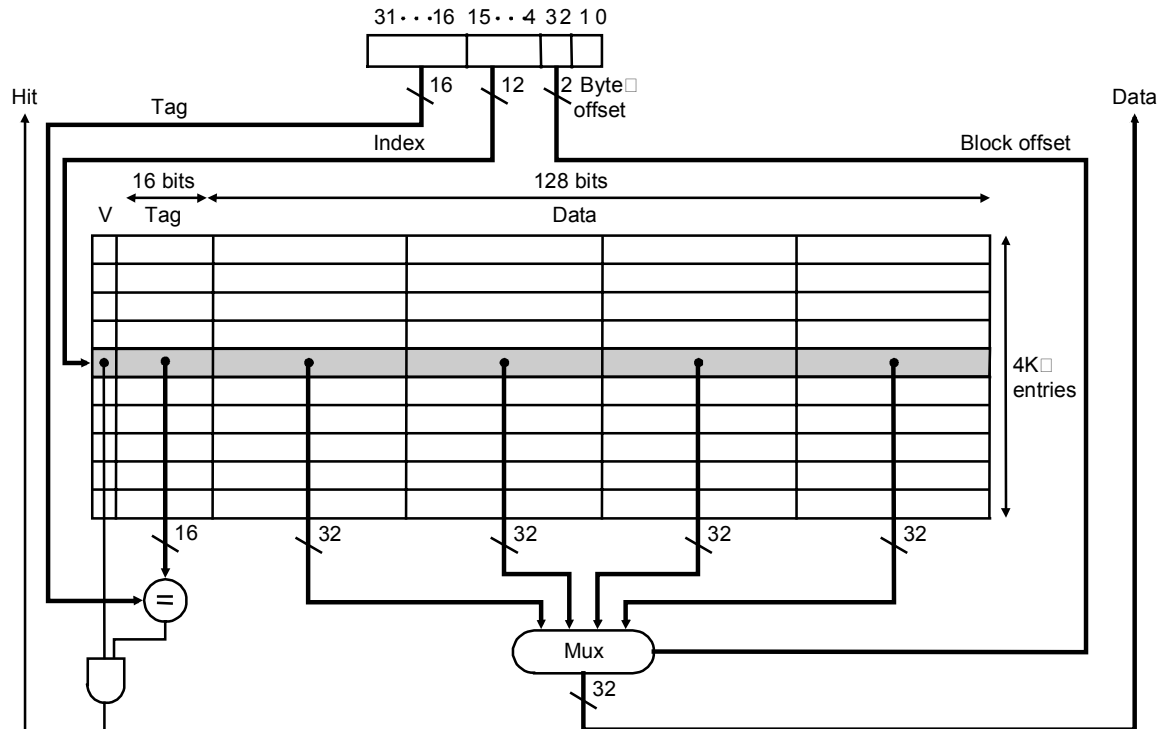
Increased Block-Length

- Instead of bringing in one word to cache when a miss occurs, bring a block of words (typically 2-4).
- Especially with instruction read, it is likely that the subsequent words will be used.
- Reduce miss rate
- Disadvantage: increase miss penalty

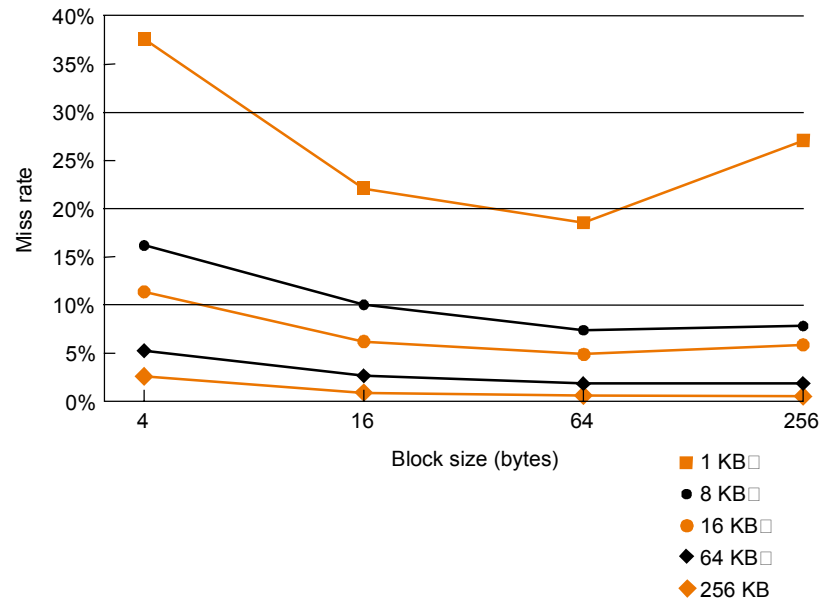
Program	Block size	I miss rate	D miss rate	Combined rate
gcc	1	6.1%	2.1%	5.4%
	4	2.0%	1.7%	1.9%
spice	1	1.2%	1.3%	1.2%
	4	0.3%	0.6%	0.4%

Mapping an Address to a Multiword Cache Block

- (Block address) mod (Number of cache blocks)
- Range: $l = \lfloor \text{Byte address} / \text{Bytes per block} \rfloor \times \text{Bytes per block}$
 $r = l + (\text{Bytes per block} - 1)$



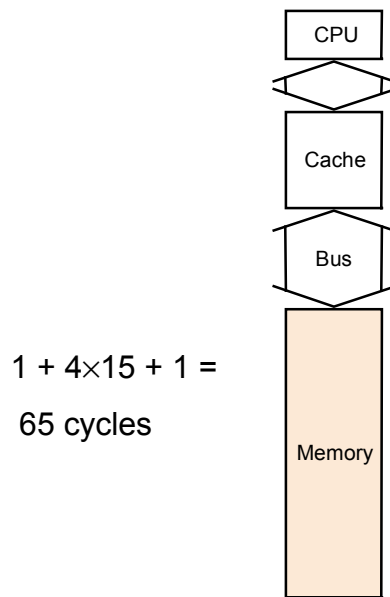
Miss Rate vs. Block Size



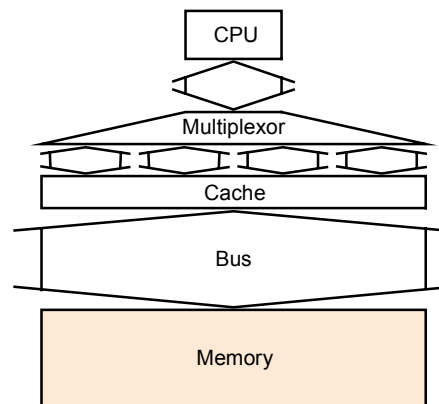
Data for a 1KB cache with different block sizes

Designing the Memory System to Support Cache

- Increase memory bandwidth to support larger block sizes
- Assume 1 cycle to send address, 15 cycles for each access initiated, 1 cycle to send word of data. Blocksize = 4 words.

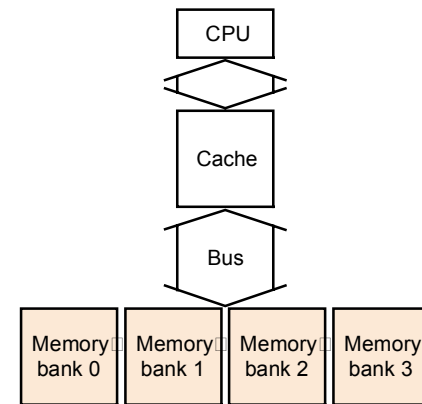


a. One-word-wide
memory organization



b. Wide memory organization

$$1 + 15 + 1 = 17 \text{ cycles}$$



c. Interleaved memory organization

$$1 + 15 + 4 \times 1 = 22 \text{ cycles}$$

Measuring Cache Performance

- **CPU time = (CPU execution clock cycles + Memory stall clock cycles) × Clock-cycle time**
 - **Memory stall clock cycles = Read-stall cycles + Write-stall cycles**
 - **Read-stall cycles = Reads/program × Read miss rate × Read miss penalty**
 - **Write-stall cycles = (Writes/program × Write miss rate × Write miss penalty) + Write buffer stalls**
- (assumes write-through cache)**
- **Write buffer stalls should be negligible and write and read miss penalties equal (cost to fetch block from memory)**
 - **Memory stall clock cycles = Mem access/program × miss rate × miss penalty**

Calculation I

- **Assume I-miss rate of 2% (gcc) and D-miss rate of 4%**
- **Assume CPI = 2 (without stalls) and miss penalty of 40 cycles**
- **Assume 36% loads/stores**
- **What is the CPI with memory stalls?**
- **How much faster would a machine with perfect cache run?**
- **What happens if the processor is made faster, but the memory system stays the same (e.g. reduce CPI to 1)?**
- **How does Amdahls's law come into play?**

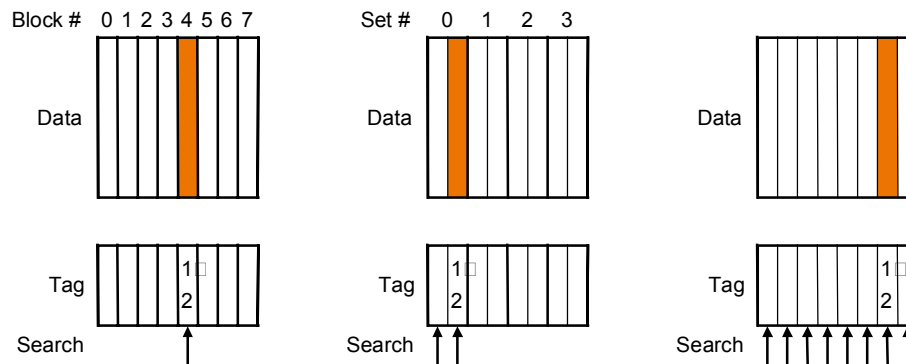
Calculation II

- **Suppose the performance of the machine in the previous example is improved by doubling the clock speed (main memory speed remains the same). Hint: since the clock rate is doubled and the memory speed remains the same, the miss penalty becomes twice as much (80 cycles).**
- **How much faster will the machine be assuming the same miss rate as the previous example?**
- **Conclusion: Relative cache penalties increase as the machine becomes faster.**

Reducing Cache Misses with a more Flexible Replacement Strategy

- In a direct mapped cache a block can go in exactly one place in cache
- In a fully associative cache a block can go anywhere in cache
- A compromise is to use a set associative cache where a block can go into a fixed number of locations in cache, determined by:

$(\text{Block number}) \bmod (\text{Number of sets in cache})$



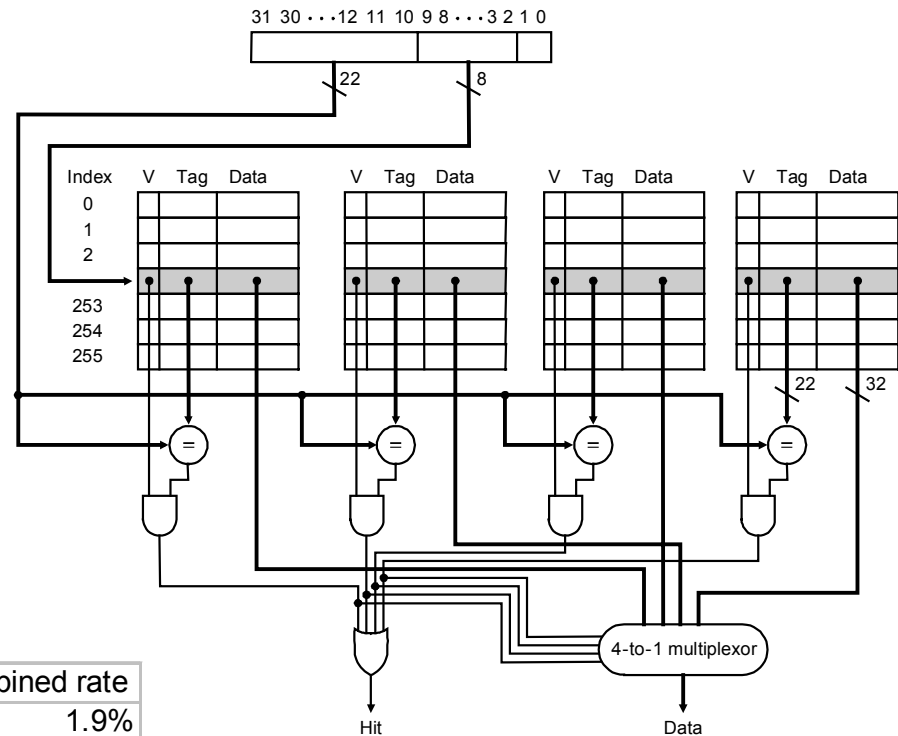
Example

- **Three small 4 word caches:
Direct mapped, two-way set associative, fully associative**
- **How many misses in the sequence of block addresses:
0, 8, 0, 6, 8?**
- **How does this change with 8 words, 16 words?**

Locating a Block in Cache

- Check the tag of every cache block in the appropriate set
- Address consists of 3 parts

tag	index	block offset
-----	-------	--------------
- Replacement strategy:
E.G. Least Recently Used (LRU)



Program	Assoc.	I miss rate	D miss rate	Combined rate
gcc	1	2.0%	1.7%	1.9%
	2	1.6%	1.4%	1.5%
	4	1.6%	1.4%	1.5%

Size of Tags vs. Associativity

- **Increasing associativity requires more comparators, as well as more tag bits per cache block.**
- **Assume a cache with 4KB blocks and 32 bit addresses**
- **Find the total number of sets and the total number of tag bits for a**
 - **direct mapped cache**
 - **two-way set associative cache**
 - **four-way set associative cache**
 - **fully associative cache**

Reducing the Miss Penalty using Multilevel Caches

- To further reduce the gap between fast clock rates of CPUs and the relatively long time to access memory additional levels of cache are used (level two and level three caches).
- The design of the primary cache is driven by the goal of a fast hit rate, which implies a relatively small size
- A secondary cache is provided to reduce the miss rate and penalty needed to go to memory.
- **Example:**
 - Assume $CPI = 1$ (with no misses) and 500 MHz clock
 - 200 ns memory access time
 - 5% miss rate for primary cache
 - secondary cache with 20 ns access time and miss rate of 2%
 - What is the total CPI with and without secondary cache?
 - How much of an improvement does secondary cache provide?