Review: (Write on the board before class)
1) Processor Bigot View so far: ISA
2) CPU: microcode => pipelined execution
3) Memory System

Break: Books are there; picked up?
Quiz solutions posted on door
meet the students, pizza and beer, Wednesday March 21 11:30 - 1:30
Project Meeting Monday 8:30-12(15) 1-2(4); Survey Friday
Assignment: 8.1, 8.3, 8.4

Today: Memory Hierarchy through First Cache idesa

General Principles
Spatial Locality: referenced again soon
Temporal Locality: nearby items referenced soon

Locality + smaller HW is faster = memory hierarchy

levels: each smaller, faster, more expensive/byte than level below
inclusive: data found in top also found in the bottom

General Principles same independent of level, but names change in practice

Definitions
Upper is closer to processor
block: minimum unit that present or not in upper level
Hit rate: fraction found in that level
So high that usually talk about *Miss rate*

Hit time: time to access upper level, including hit determination
Miss penalty: time ro replace a block from lower level, including time to replace in CPU
access time: time to lower level
= f(lower level latency)
transfer time: time to transfer block
=f(BW upper & lower)
Block frame address + block offset address

Miss rate fallacy as architecture to HW in CPU, miss rate to average memory access time in memory

$$\text{Average memory-access time} = \text{Hit time} + \text{Miss rate} * \text{Miss penalty}$$

clocks or time (ns)

See Figures 8.3 and 8.4

Implications to CPU:
non-predicatable memory access time
10s of clock cycles: wait
1000s of clock cyles: interrupt & switch & do something else

Also: how handle miss (10s => HW, 1000s => SW)

Also: fast hit check since every memory access

All levels of memory hierarchy ask 4 questions:
Q1: Where can a block be placed in the upper level? (*Block placement*)
Q2: How is a block found if it is in the upper level? (*Block identification*)
Q3: Which block should be replaced on a miss? (*Block replacement*)
Q4: What happens on a write? (*Write strategy*)

Caches: = “buffer”
Answer questions:
Q1: Where put it (block placement)
DM,FA,SA (n-way => n blocks in a set)
DM = block number mod number of blocks => index
See 8.6
Q2: How find it (block ID)
TAg IDs the block: placement determines the search
See 8.7
How big is the tag?
Q3: Which Replaced?
easy if cache empty, but almost always full
DM: no choice
SA or FA
Random or
LRU: keep track of furthest away
Q4: Write?

Little difference for caches in first 3 Qs
Most accesses are reads (9% stores + 17% loads)
Read data and tag same time

write Change a portion of block
2 choices
Write through
WRite back
clean or dirty
Pros and Cons of each?
WT: read misses cannot result in writes
WB: no writes of repeated writes
WT: stall until memory complete
=> Write Buffers

2 options
write allocate: allocate on a miss
no write allocate: just write memory

Example: VAX-11/780 cache Figure 8.11
Mux not needed for DM

Performance

$$\text{CPU time} = (\text{CPU-execution clock cycles} + \text{Memory-stall clock cycles}) * \text{Clock cycle time}$$

$$\text{Memory-stall clock cycles} = \frac{\text{Reads}}{\text{Program}} * \text{Read miss rate} * \text{Read miss penalty}$$

$$+ \frac{\text{Writes}}{\text{Program}} * \text{Write miss rate} * \text{Write miss penalty}$$

$$\text{Memory-stall clock cycles} = \frac{\text{Memory accessess}}{\text{Program}} * \text{Miss rate} * \text{Miss penalty}$$

$$\text{CPU time} = \text{IC} * \left(\text{CPI}_{\text{Execution}} + \frac{\text{Memory accesses}}{\text{Instruction}} * \text{Miss rate} * \text{Miss penalty}\right) * \text{Clock cycle time}$$

3 Cs
Reason for misses
8.13

Compulsory (cold start or first refernence)
Capacity
Conclict

Examples
Block Size
8.14/8.15

Instruction vs. Data vs. Unified
8.16

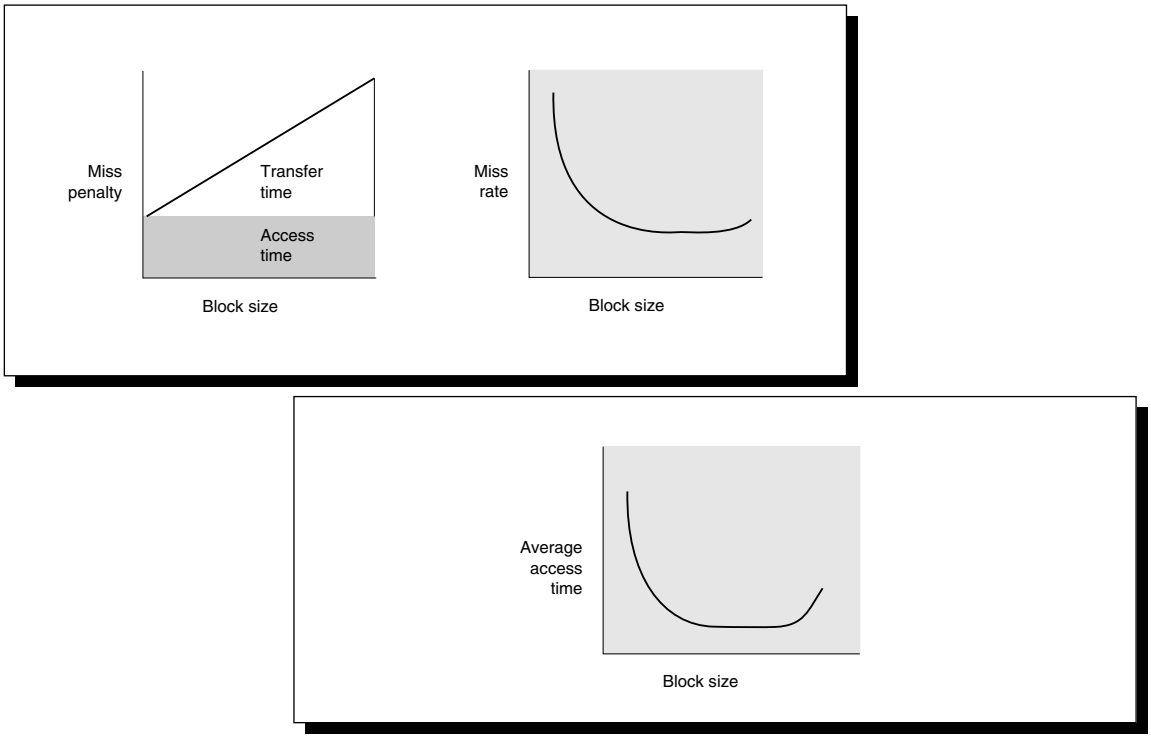

FIGURE 8.3 Block size versus miss penalty and miss rate. The transfer-time portion of the miss penalty obviously grows with increasing block size. For a fixed-size upper-level memory, miss rates fall with increasing block size until so much of the block is not used that it displaces useful information in the upper level, and miss rates begin to rise. The point on the curve on the right where miss rates begin to rise with increasing block size is sometimes called the *pollution point.* **FIGURE 8.4** The relationship between average memory-access time and block size.

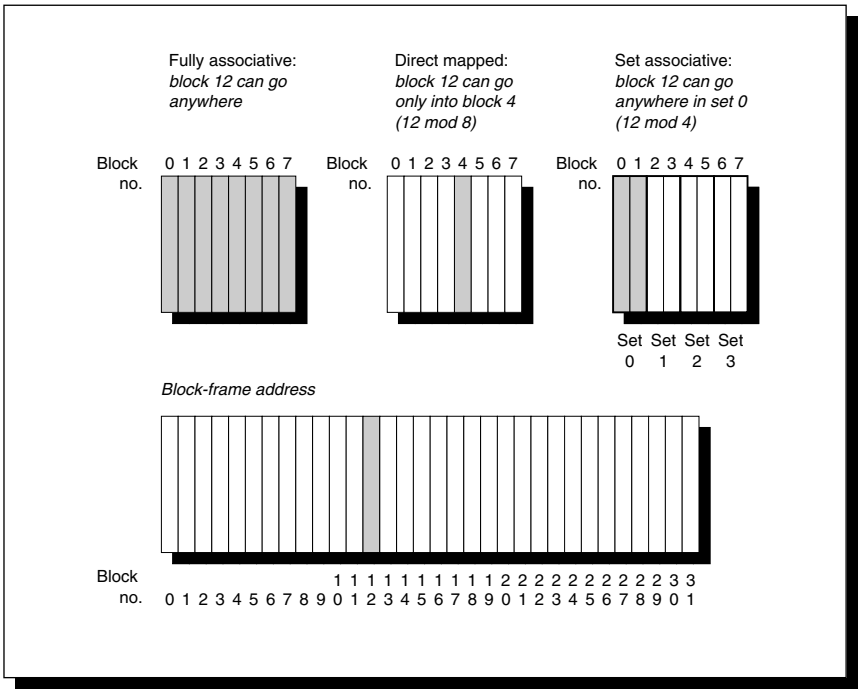

FIGURE 8.6 The cache has 8 blocks, while memory has 32 blocks. The set-associative organization has 4 sets with 2 blocks per set, called two-way set associative. (Real caches contain hundreds of blocks and real memories contain hundreds of thousands of blocks.) Assume that there is nothing in the cache and that the block-frame address in question identifies lower-level block 12. The three options for caches are shown left to right. In fully associative, block 12 from the lower level can go into any of the 8 blocks of the cache. With direct mapped, block 12 can only be placed into block 4 (12 modulo 8). Set associative, which has some of both features, allows the block to be placed anywhere in set 0 (12 modulo 4). With two blocks per set, this means block 12 can be placed either in block 0 or block 1 of the cache.

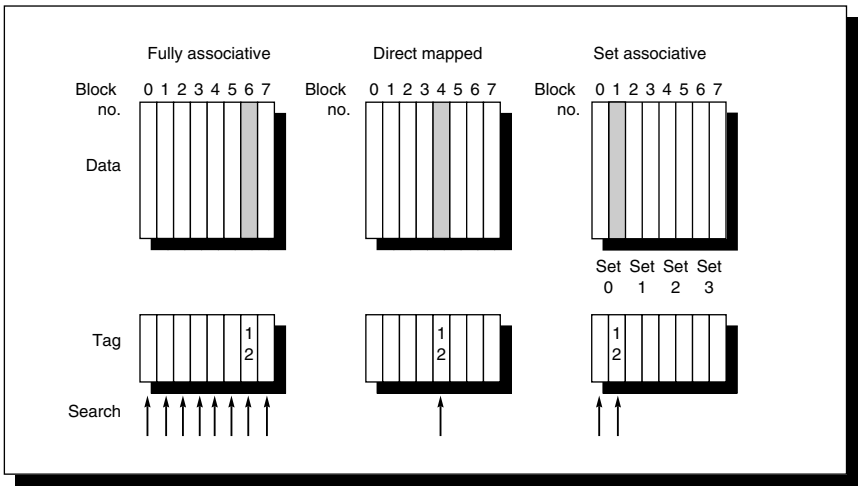

FIGURE 8.7 In fully associative placement, the block for block-frame address 12 can appear in any of the 8 blocks; thus, all 8 tags must be searched. The desired data is found in cache block 6 in this example. In direct-mapped placement there is only one cache block where memory block 12 can be found. In set-associative placement, with 4 sets, memory block 12 must be in set 0 (12 mod 4); thus, the tags of cache blocks 0 and 1 are checked. In this case the data is found in cache block 1. Speed of cache access dictates that searching must be performed in parallel for fully associative and set-associative mappings.

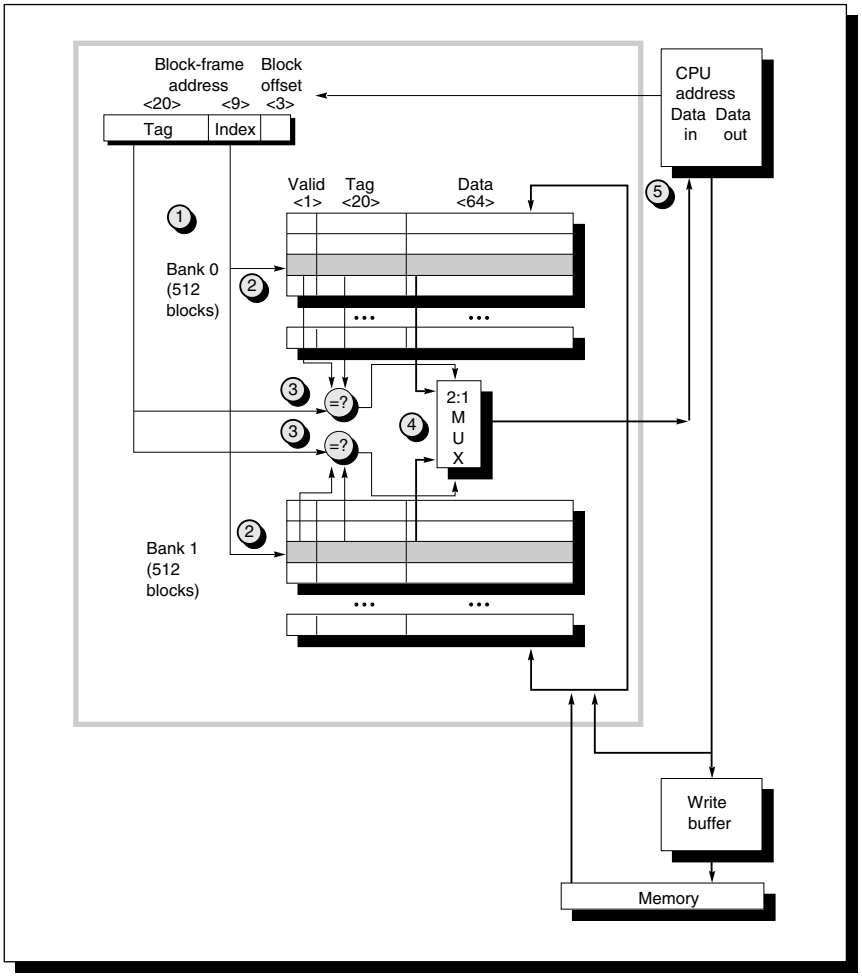

FIGURE 8.11 The organization of the VAX-11/780 cache. The 8-KB cache is two-way set associative with 8-byte blocks. It has 512 sets with two blocks per set; the set is selected by the 9-bit index. The five steps of a read hit, shown as circled numbers in order of occurrence, label this organization. The line from memory to the cache is used on a miss to load the cache. Such multiplexing is not needed in a direct-mapped cache. Note that the offset is connected to chip select of the data SRAMs to allow the proper words to be sent to the 2:1 multiplexer.

$$\text{CPU time} = (\text{CPU-execution clock cycles} + \text{Memory-stall clock cycles}) * \text{Clock cycle time}$$

$$\text{Memory-stall clock cycles} = \frac{\text{Reads}}{\text{Program}} * \text{Read miss rate} * \text{Read miss penalty}$$

$$+ \frac{\text{Writes}}{\text{Program}} * \text{Write miss rate} * \text{Write miss penalty}$$

$$\text{Memory-stall clock cycles} = \frac{\text{Memory accessess}}{\text{Program}} * \text{Miss rate} * \text{Miss penalty}$$

$$\text{CPU time} = \text{IC} * \left(\text{CPI}_{\text{Execution}} + \frac{\text{Memory accesses}}{\text{Instruction}} * \text{Miss rate} * \text{Miss penalty} \right) * \text{Clock cycle time}$$

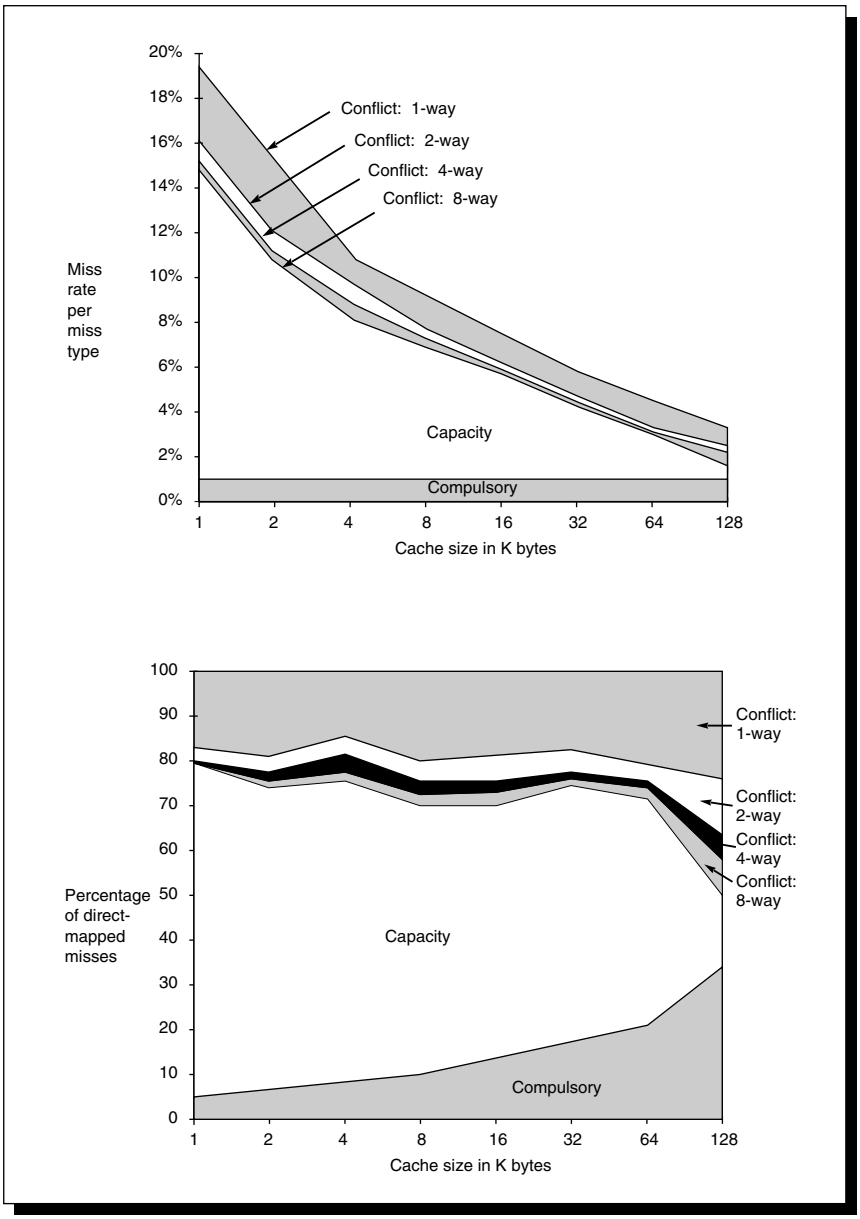

FIGURE 8.13 Total miss rate (top) and distribution of miss rate (bottom) for each size cache according to three Cs for the data in Figure 8.12 (page 421). The top diagram is the actual miss rates, while the bottom diagram is scaled to the direct-mapped miss ratio.

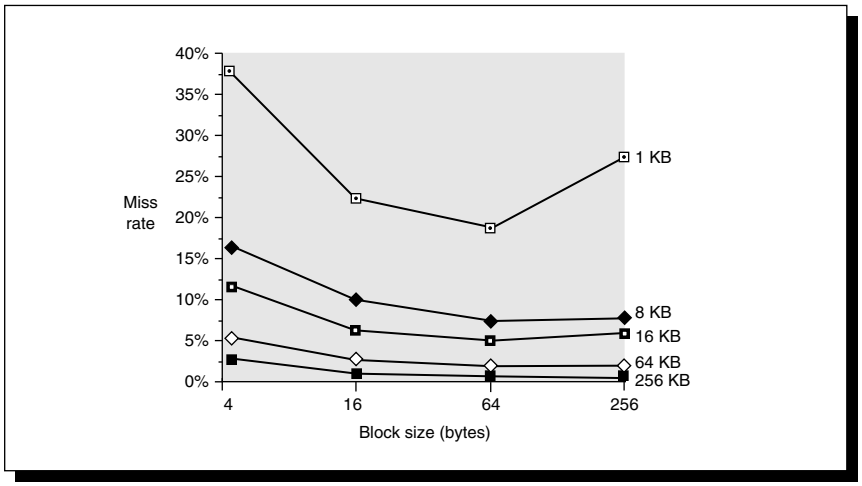

FIGURE 8.14 Miss rate versus block size. Note that for a 1-KB cache, 256-byte blocks have a higher miss rate than either 16- or 64-byte blocks. (The smallest block is 4 bytes.) In this particular example, the cache would have to be 256 KB in order for increasing block size to always result in decreased misses.

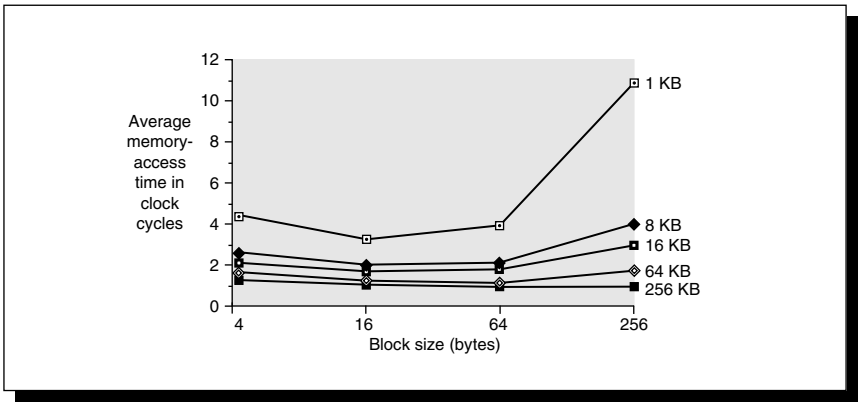

FIGURE 8.15 Average access time versus block size using the miss rates in Figure 8.14. This assumes an 8–clock-cycle latency and that the memory and bus can transfer 4 bytes per clock cycle. On a miss all the blocks are loaded into the cache before the requested word is sent to the CPU. The lowest average memory-access time is either for 16-byte or 64-byte blocks, and 256-byte blocks are better than 4-byte blocks only for the largest cache.

Size	Instruction only	Data only	Unified
0.25 KB	22.2%	26.8%	28.6%
0.50 KB	17.9%	20.9%	23.9%
1 KB	14.3%	16.0%	19.0%
2 KB	11.6%	11.8%	14.9%
4 KB	8.6%	8.7%	11.2%
8 KB	5.8%	6.8%	8.3%
16 KB	3.6%	5.3%	5.9%
32 KB	2.2%	4.0%	4.3%
64 KB	1.4%	2.8%	2.9%
128 KB	1.0%	2.1%	1.9%
256 KB	0.9%	1.9%	1.6%

FIGURE 8.16 Miss rates for instruction-only, data-only, and unified caches of different sizes. The data are for a 2-way–associative cache using LRU replacement with 16-byte blocks for an average of user/system traces on the VAX-11 and system traces on the IBM 370 [Hill 1987]. The percentage of instruction references in these traces is about 53%.