

Lecture 23

Multilevel Caches

School of Computer Science and Engineering
Soongsil University

5. Large and Fast: Exploiting Memory Hierarchy

- 5.1 Introduction
- 5.2 Memory Technologies
- 5.3 The Basics of Caches
- 5.4 Measuring and Improving Cache Performance
- 5.5 Dependable Memory Hierarchy
- 5.6 Virtual Machines
- 5.7 Virtual Memory
- 5.8 A Common Framework for Memory Hierarchies
- 5.9 Using a Finite-State Machine to Control a Simple Cache
- 5.10 Parallelism and Memory Hierarchies: Cache Coherence
- 5.11 Parallelism and Memory Hierarchy: RAID
- 5.12 Advanced Material: Implementing Cache Controllers
- 5.13 Real Stuff: The ARM Cortex-A8 and Intel Core i7 Memory Hierarchies

Choosing Which Block to Replace

- **Replacement algorithms**

- ❖ Direct-mapped cache ... uniquely specified
- ❖ Set-associative cache ... choice among the blocks in the set
- ❖ Fully associative cache ... choice among all the blocks

- **LRU (least recently used)**

- ❖ Victim ... the block that has been unused for the longest time
- ❖ Implementation
 - by keeping track of when each element was used
- ❖ For 2-way set-associative cache
 - 1 reference bit
- ❖ Higher degree of associativity
 - Harder to implement

Example: Tag size vs. Associativity (1)

- 32-bit address, cache of 4096 ($=2^{12}$) bytes with block size = 1 byte
- Direct, 2-way, 4-way, fully associative

[Answer]

- ❖ Direct mapped
 - 4K blocks \rightarrow 12-bit index
 - tag = $(32 - 12) \times 4K = 80K$ bits
- ❖ 2-way set associative
 - 2K sets \rightarrow 11-bit index
 - tag = $(32 - 11) \times 2 \times 2K = 84K$ bits
- ❖ 4-way set associative
 - 1K sets \rightarrow 10-bit index
 - tag = $(32 - 10) \times 4 \times 1K = 88K$ bits
- ❖ Fully associative
 - 1 set, tag = $32 \times 4K \times 1 = 128K$ bits

Example: Tag size vs. Associativity (2)

- Cache of 4096 ($=2^{12}$) bytes with block size = 4 words

[Answer]

- ❖ Number of blocks = $4\text{KB}/16\text{B} = 2^8 = 256$
- ❖ 16 bytes per block \rightarrow 4-bit block offset
- ❖ Direct mapped
 - 256 blocks \rightarrow 8-bit index
 - tag = $(28 - 8) \times 256 = 5\text{K bits}$
- ❖ 2-way set associative
 - 128 sets \rightarrow 7-bit index
 - tag = $(28 - 7) \times 2 \times 128 = 5.25\text{K bits}$
- ❖ 4-way set associative
 - 64 sets \rightarrow 6-bit index
 - tag = $(28 - 6) \times 4 \times 64 = 5.5\text{K bits}$
- ❖ Fully associative
 - 1 set, tag = $28 \times 256 \times 1 = 7\text{K bits}$

Example: Tag size vs. Associativity (3)

- Cache of 4096 ($=2^{12}$) blocks with block size = 4 words

[Answer]

- ❖ Cache size = 4K x 16 bytes = 64K bytes
- ❖ 16 bytes per block -> 4-bit offset
- ❖ Direct mapped
 - 4K sets -> 12-bit index
 - tag = $(28 - 12) \times 4K = 64K$ bits
- ❖ 2-way set associative
 - 2K sets -> 11-bit index
 - tag = $(28 - 11) \times 2 \times 2K = 68K$ bits
- ❖ 4-way set associative
 - 1K sets -> 10-bit index
 - tag = $(28 - 10) \times 4 \times 1K = 72K$ bits
- ❖ Fully associative
 - 1 set, tag = $28 \times 4K \times 1 = 112K$ bits

Reducing the Miss Penalty Using Multilevel Caches

- **Primary cache**

- ❖ On-chip

- **Secondary cache**

- ❖ Off-chip SRAMs
- ❖ Reduce primary cache's miss penalty

- **Example : Performance of Multilevel Caches**

- ❖ Clock rate = 4 GHz
- ❖ CPI = 1.0 with a primary cache of 100% hit rate
- ❖ Main memory access time = 100 ns (including miss handling)
- ❖ Miss rate/instruction at the primary cache = 2%
- ❖ **With a secondary cache below, how will it be faster?**
global miss rate = 0.5%, access time = 5 ns

[Answer]

❖ With 1 level of cache

Miss penalty to main memory

$$= 100\text{ns}/0.25\text{ns} = 400 \text{ clock cycles}$$

Total CPI

$$= \text{base CPI} + \text{memory-stall cycles per instruction}$$

$$= 1.0 + 2\% \times 400 = 9.0$$

❖ With 2 levels of caches

Miss penalty to secondary cache

$$= 5\text{ns}/0.25\text{ns} = 20 \text{ clock cycles}$$

Total CPI = 1 + primary stalls + secondary stalls

$$= 1 + 2\% \times 20 + 0.5\% \times 400 = 3.4$$

❖ Thus, speedup = $9.0/3.4 = 2.6$

Design Considerations for a Primary and Secondary Cache

■ Primary caches

- ❖ Focus on **minimizing hit time**
 - ◆ to yield a shorter clock cycle or fewer pipeline stages
- ❖ Smaller cache
- ❖ Smaller block size → reduced miss penalty

■ Secondary caches

- ❖ Focus on **reducing miss rate**
 - ◆ to reduce the penalty of long memory access times
- ❖ Larger
 - ◆ since access time is less critical
- ❖ Larger block size and higher associativity
 - ◆ to reduce miss rates

Caches of Real Machines

	Intel Nehalem	AMD Barcelona
L1 cache organization & size	Split I\$ and D\$; 32KB for each per core; 64B blocks	Split I\$ and D\$; 64KB for each per core; 64B blocks
L1 associativity	4-way (I), 8-way (D) set assoc.; ~LRU replacement	2-way set assoc.; LRU replacement
L1 write policy	write-back, write-allocate	write-back, write-allocate
L2 cache organization & size	Unified; 256KB (0.25MB) per core; 64B blocks	Unified; 512KB (0.5MB) per core; 64B blocks
L2 associativity	8-way set assoc.; ~LRU	16-way set assoc.; ~LRU
L2 write policy	write-back, write-allocate	write-back, write-allocate
L3 cache organization & size	Unified; 8192KB (8MB) shared by cores; 64B blocks	Unified; 2048KB (2MB) shared by cores; 64B blocks
L3 associativity	16-way set assoc.	32-way set assoc.; evict block shared by fewest cores
L3 write policy	write-back, write-allocate	write-back; write-allocate

Summary: Improving Cache Performance

1. Reduce the time to hit in the cache

- ❖ smaller cache
- ❖ direct mapped cache
- ❖ smaller blocks
- ❖ for writes
 - ◆ no write allocate – no “halt” on cache, just write to write buffer
 - ◆ write allocate – to avoid two cycles (first check for hit, then write) pipeline writes via a delayed write buffer to cache

2. Reduce the miss rate

- ❖ bigger cache
- ❖ more flexible placement (increase associativity)
- ❖ larger blocks (16 to 64 bytes typical)
- ❖ victim cache – small buffer holding most recently discarded blocks

Summary: Improving Cache Performance

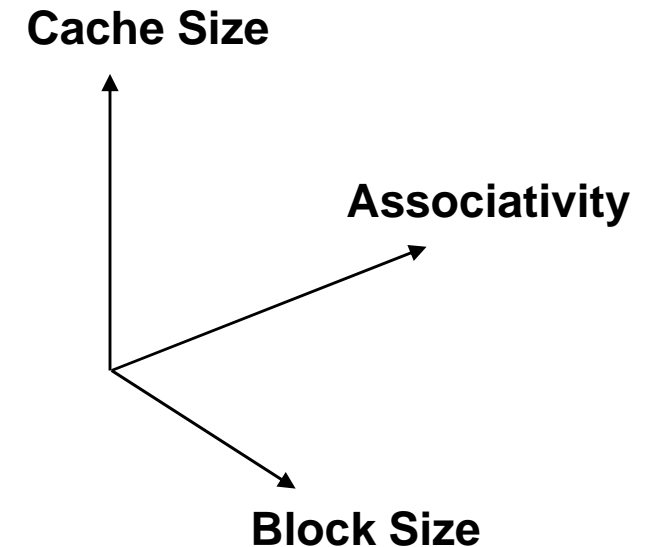
3. Reduce the miss penalty

- ❖ smaller blocks
- ❖ use a write buffer to hold dirty blocks being replaced so don't have to wait for the write to complete before reading
- ❖ check write buffer (and/or victim cache) on read miss – may get lucky
- ❖ for large blocks fetch critical word first
- ❖ use multiple cache levels – L2 cache not tied to CPU clock rate
- ❖ faster backing store/improved memory bandwidth
 - ◆ wider buses
 - ◆ memory interleaving, DDR SDRAMs

Summary: The Cache Design Space

■ Several interacting dimensions

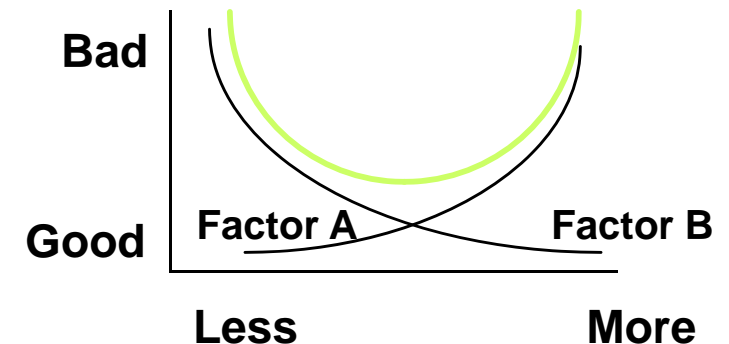
- ❖ cache size
- ❖ block size
- ❖ associativity
- ❖ replacement policy
- ❖ write-through vs write-back
- ❖ write allocation



■ The optimal choice is a compromise

- ❖ depends on access characteristics
 - ◆ workload
 - ◆ use (I-cache, D-cache, TLB)
- ❖ depends on technology / cost

■ Simplicity often wins



5.5 Dependable Memory Hierarchy

- **RAS (Reliability, Availability, Serviceability)**
- **Dependability**
 - ❖ “a measure of a system's availability, reliability, and its maintainability” from Wikipedia

Defining Failure

- **Two states of delivered service with respect to the service specification**
 1. Service accomplishment
 2. Service interruption
- **State transition**
 - ❖ Failure: state 1 -> state 2
 - ❖ Restoration: state 2 -> state 1

Reliability

- **Reliability**

- ❖ Measure of the continuous service accomplishment

- **MTTF (Mean Time To Failure)**

- **Annual failure rate (AFR)**

- ❖ Percentage of devices that would be expected to fail in a year for a given MTTF

- **MTBF (Mean Time Between Failure)**

- = MTTF + MTTR (Mean Time To Repair)**

Availability

■ Availability

- ❖ Measure of the service accomplishment with respect to the alteration between the two states

$$\text{Availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

■ How to reduce MTTR

- ❖ Improved tools and processes for diagnosis and repair

■ How to improve MTTF

1. Fault avoidance: Preventing fault occurrence by construction.
2. Fault tolerance: Using redundancy to allow the service to comply with the service specification despite faults occurring.
3. Fault forecasting: Predicting the presence and creation of faults, allowing the component to be replaced before it fails.

The Hamming SEC/DED Code

■ **Richard Hamming**

- ❖ Invented a popular redundancy scheme for memory, for which he received the Turing Award in 1968

■ **Hamming distance**

- ❖ Minimum number of bits that are different between any two correct bit patterns
- ❖ Hamming distance = 2 provides single bit error detection
 - ◆ E.g. parity code
- ❖ Hamming distance = 3 provides single error correction/double error detection

■ **Parity code**

- ❖ Single error detection
- ❖ Even/Odd parity ... The number of 1s in a word is even/odd
- ❖ 1-bit parity code cannot detect 2 bits of error
- ❖ 1-bit parity code cannot correct errors

Hamming Error Correction Code (ECC)

Bit position		1	2	3	4	5	6	7	8	9	10	11	12
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8
Parity bit coverage	p1	X		X		X		X		X		X	
	p2		X	X			X	X			X	X	
	p4				X	X	X	X					X
	p8								X	X	X	X	X

Figure 5.24

■ Encoding SEC

- ❖ All bit positions that are a power 2 are parity bits
- ❖ Even parity

■ Decoding SEC

- ❖ Value of parity bits indicates which bits are in error
 - ◆ Parity bits = 0000 indicates no error
 - ◆ Parity bits = 1001 indicates error in bit 9

Example: Hamming Error Correction Code

- **Data = 9A_{hex} = 10011010_{two}**
 - ❖ Show Hamming ECC.
 - ❖ Invert bit 10 and show that the ECC corrects the single bit error.

[Answer]

Bit position	1	2	3	4	5	6	7	8	9	10	11	12
	p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8
	0	1	1	1	0	0	1	0	1	0	1	0
p1 (xxx1)	0		1		0		1		1		1	
p2 (xx1x)		1	1			0	1			0	1	
p4 (x1xx)				1	0	0	1					0
p8 (1xxx)								0	1	0	1	0

- ❖ Invert bit 10: 011100101**0**10 -> 011100101**1**10
- ❖ p1=E, p2=**0**, p4=E, p8=**0** -> **1010**_{two}=10_{ten} -> error in bit 10_{ten}

Hamming SEC/DED Code

- **Single Error Correcting, Double Error Detecting Code**

- ❖ With one more parity bit, the minimum Hamming distance in a code = 4
- ❖ Add a parity bit that is calculated over the whole word

- **Example: data=1001**

Bit position	1	2	3	4	5	6	7	8	
	p_1	p_2	d1	p_3	d2	d3	d4	p_4	
encoding	0	0	1	1	0	0	1	1	
case 1	0	0	1	1	0	0	1	1	no error
case 2	0	0	1	1	0	0	0	1	single error (d4)
case 3	0	0	1	1	0	0	1	0	p_4 error
case 4	0	0	1	1	1	1	1	1	double error (d2,d3)

- **72-bit DIMM**

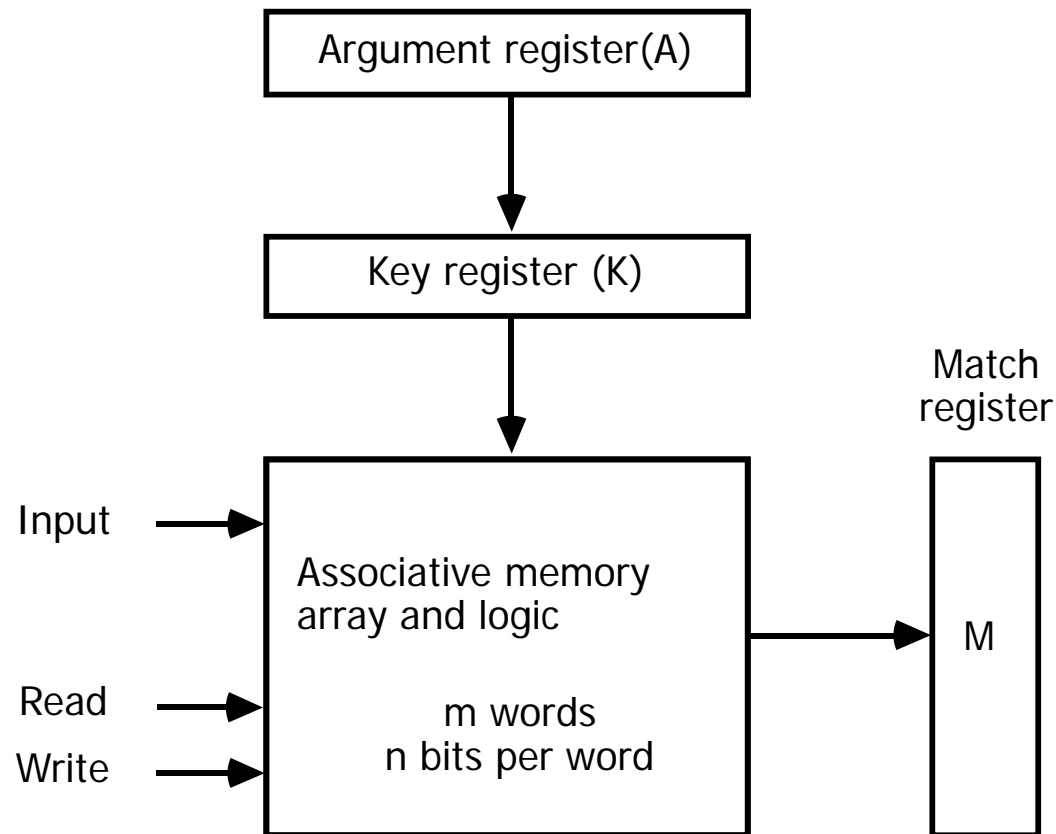
- ❖ 8-byte data block + 1-byte parity block

Supplement

Fully Associative Cache

[ref] Mano, Computer System Architecture

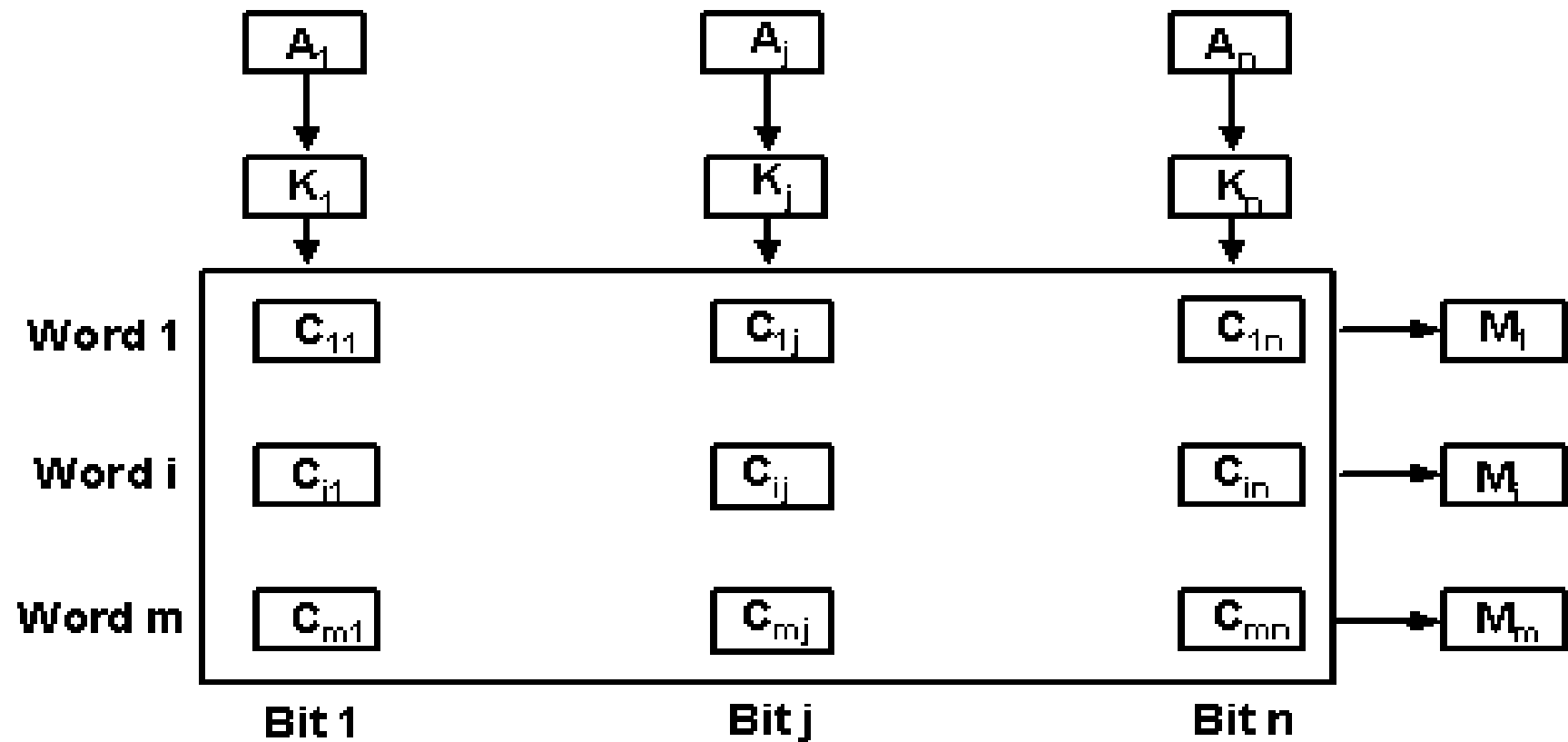
- Associative Memory = Content Addressable Memory (CAM)
 - ❖ Accessed by the content of the data rather than by an address



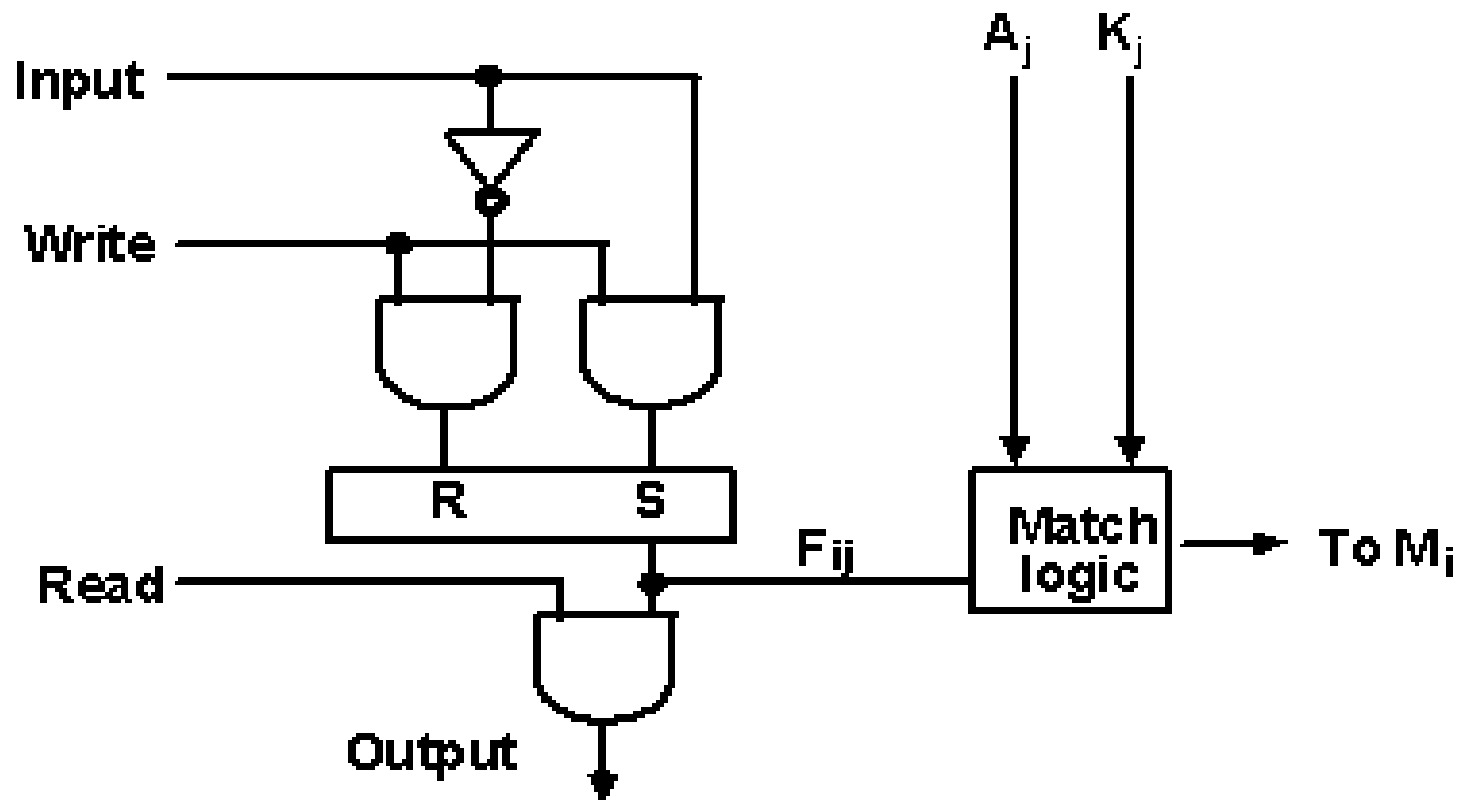
Operation of Associative Memory

- Compare each word in CAM in parallel with the contents of A (Argument Register)
- If $\text{CAM Word}[i] = A$, $M(i) = 1$
- Read sequentially accessing CAM for $M(i) = 1$
- K (Key Register) provides a mask for choosing a particular field or key in the argument in A
 - ❖ only those bits in the argument that have 1's in their corresponding position of K are compared

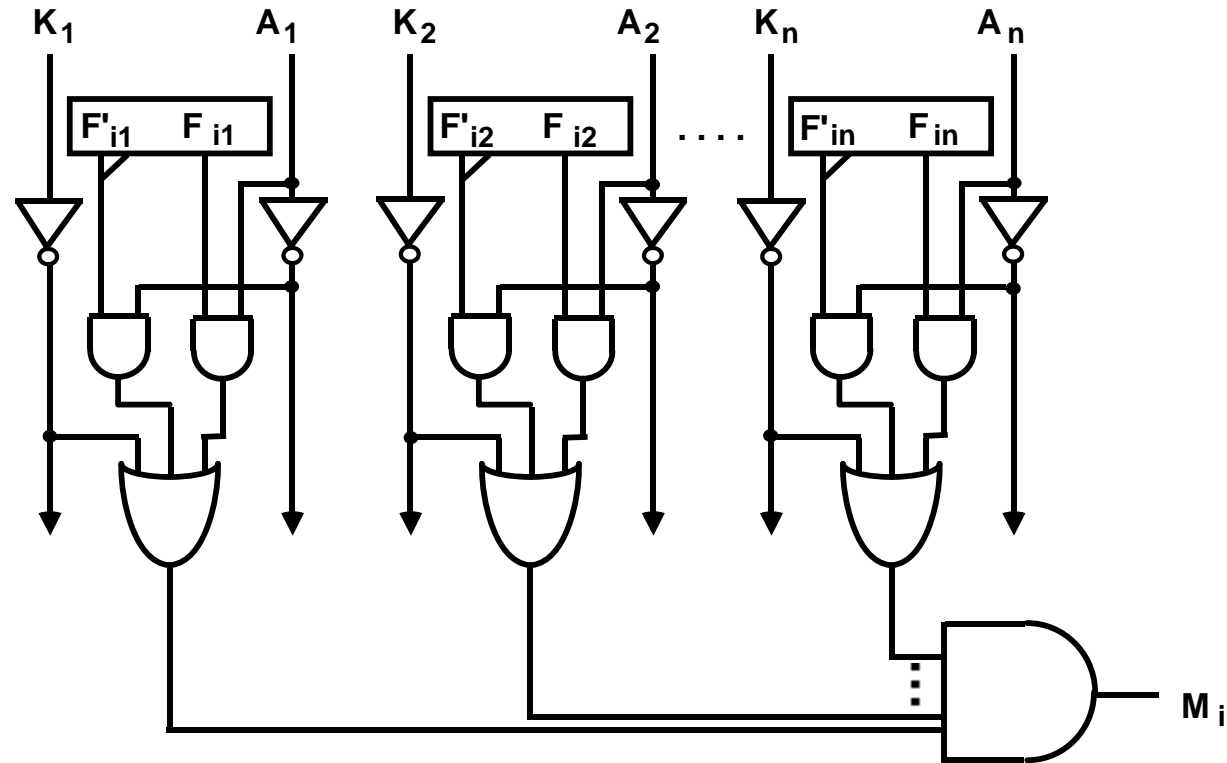
Internal Organization of CAM



Typical Cell C_{ij} of CAM



Match Logic



- $$M_i = \prod \{ (F_{ij} \cdot A_j) + (F'_{ij} \cdot A'_j) + K'_j \}$$

Understanding Program Performance

■ Impact of the memory hierarchy on performance

- ❖ LaMarca, A. and R. E. Ladner [1996]. "The influence of caches on the performance of heaps," *ACM J. of Experimental Algorithmics*, Vol. 1.

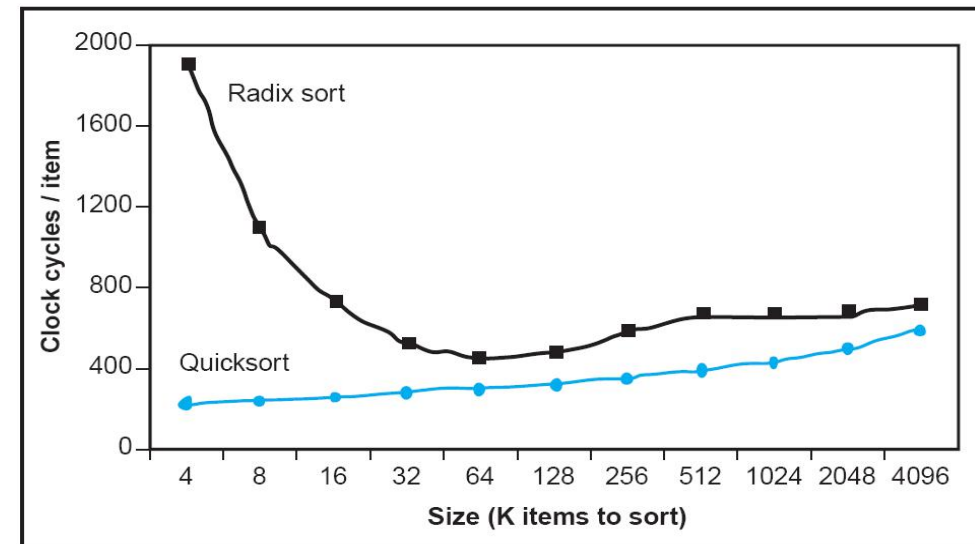
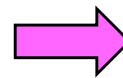
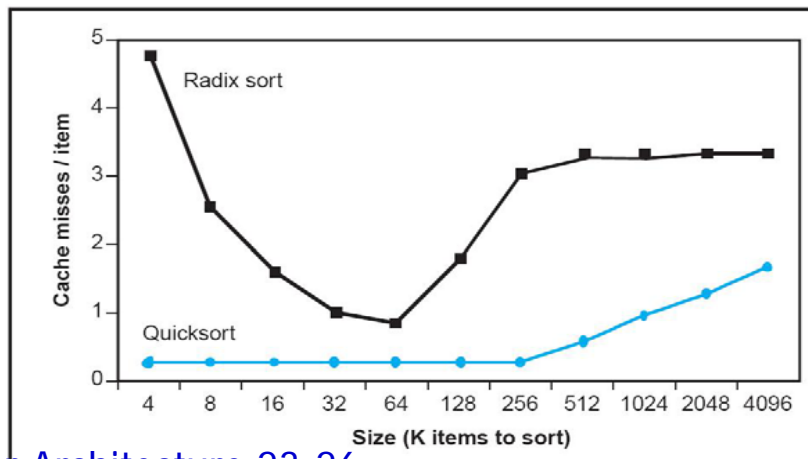
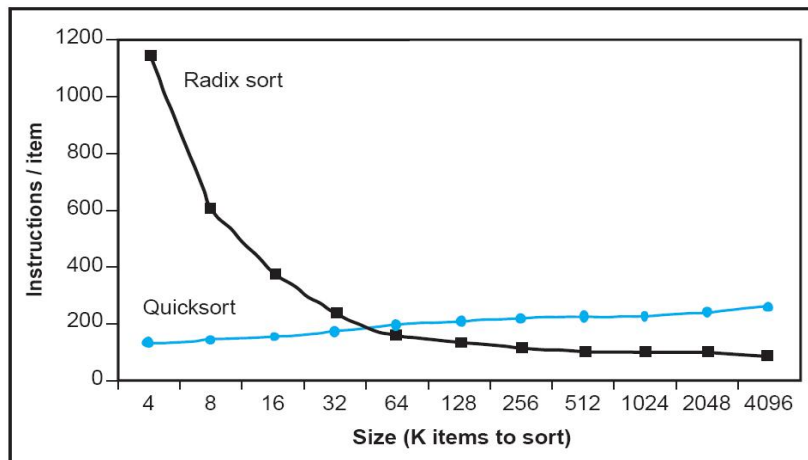


Figure 5.18

Designing the Memory System to Support Caches

- **Increased bandwidth from memory to cache**
 - ❖ Reduced miss penalty
 - ❖ Larger block size with low miss penalty
- **Bus**
 - ❖ Connecting memory to processor
 - ❖ Much slower than the processor, by as much as a factor of 10
- **Assumptions**
 - ❖ 1 memory bus clock cycle to send address
 - ❖ 15 clock cycles for each DRAM access initiated
 - ❖ 1 clock cycle to send a word of data
 - ❖ Cache block of 4 words

One-word-wide Memory

- A one-word-wide bank of DRAMs
- All accesses are made sequentially.
- Miss penalty
 - ❖ $1 + 4 \times 15 + 4 \times 1 = 65$ (clock cycles)
- The number of bytes transferred per clock cycle for a single miss
 - ❖ $(4 \times 4) / 65 = 0.25$ (bytes/cycle)
- Timing

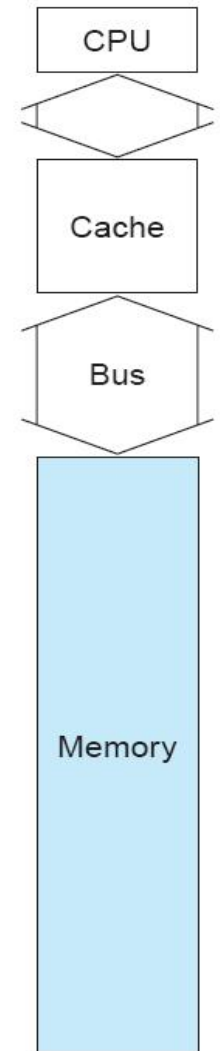
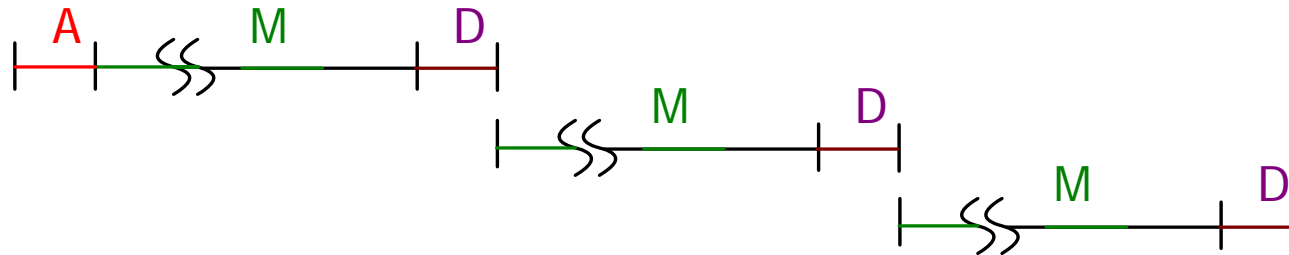


Figure 5.11a

Wide Memory (1/2)

- Increase the bandwidth to memory by widening the memory and the bus
 - ❖ Parallel access to all the words of the block
- **Major costs**
 - ❖ Wider bus
 - ❖ Potential increase in cache access time due to the multiplexor and control logic between processor and cache

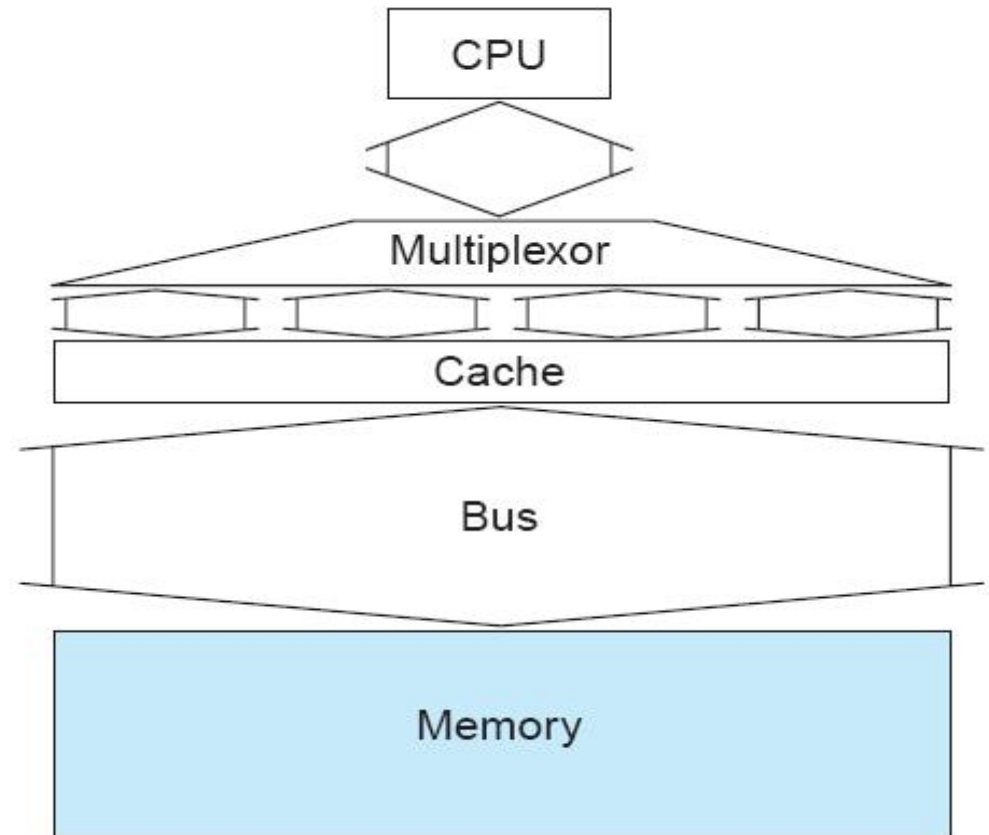
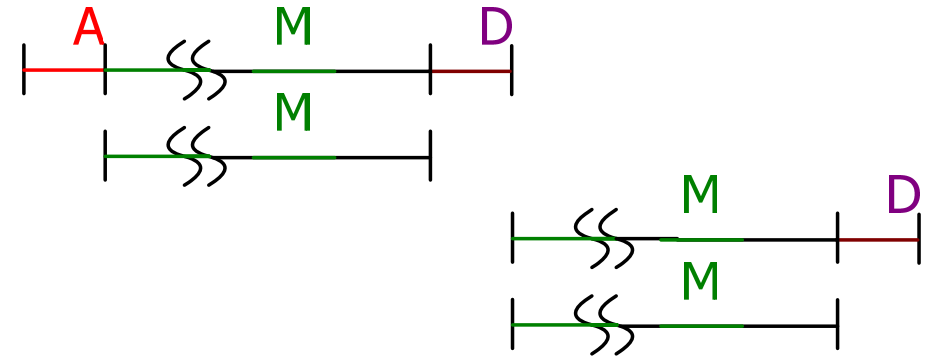


Figure 5.11b

Wide Memory (2/2)

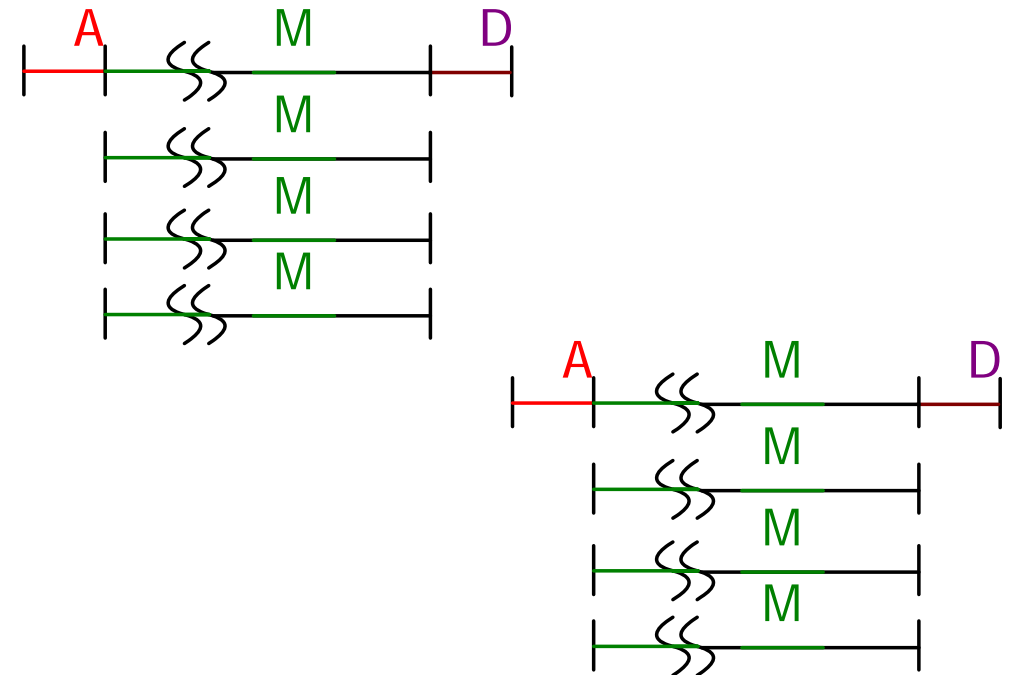
- **If a main memory width = 2 words**

- ❖ Miss penalty = $1 + 2 \times 15 + 2 \times 1 = 33$ (clock cycles)
- ❖ Bandwidth for a single miss = $4 \times 4 / 33 = 0.48$ (bytes/cycle)



- **If a main memory width = 4 words**

- ❖ Miss penalty = $1 + 1 \times 15 + 1 \times 1 = 17$ (clock cycles)
- ❖ Bandwidth for a single miss = $4 \times 4 / 17 = 0.94$ (bytes/cycle)



Interleaved Memory (2/1)

- Widening the memory but not the bus and cache
- Organize the memory chips in banks to read or write multiple words in one access time
- Advantage of incurring the full memory latency only once
- Each bank can be written independently
 - ❖ 4 times write bandwidth
 - ❖ Less stall occurs in write-through cache

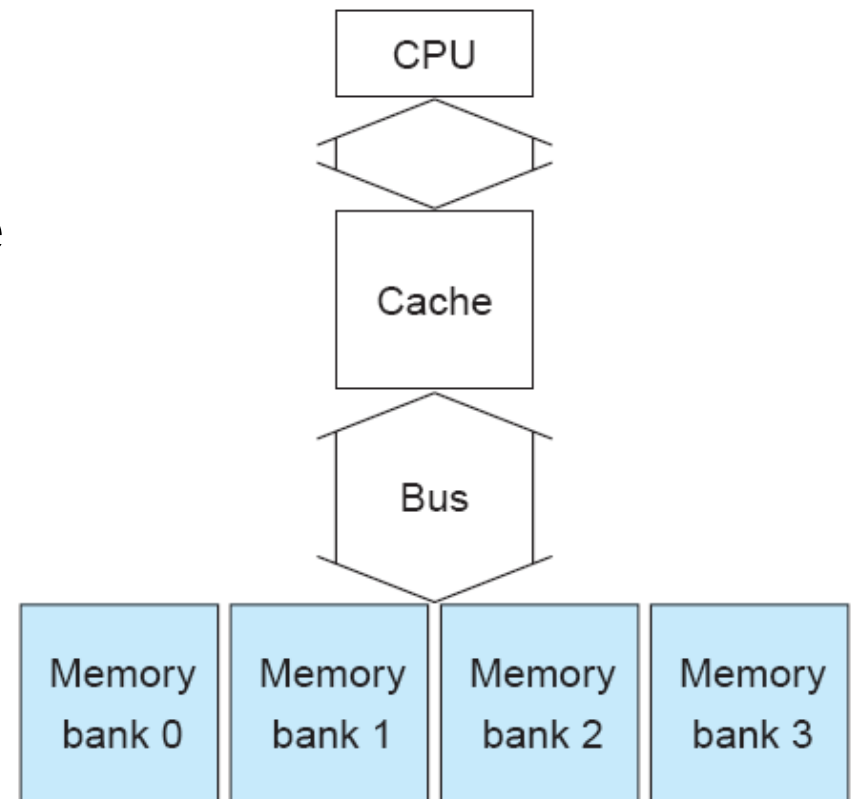


Figure 5.11c

Interleaved Memory (2/2)

■ With 4 banks

- ❖ 4-way interleaved memory
- ❖ Miss penalty = $1 + 1 \times 15 + 4 \times 1 = 20$ (clock cycles)
- ❖ Bandwidth for a single miss = $4 \times 4 / 20 = 0.80$

