University Of Diyala
College Of Engineering
Computer Engineering Department

COMPUTER ARCHITECTURE II

PART 3: CACHE PERFORMANCE OPTIMIZATION

Asst. Prof. Ahmed Salah Hameed

Second stage

2022-2023

FROM THE PREVIOUS LECTURE

CACHE ORGANIZATION (PLACEMENT POLICIES)

Fully Associative Cache

Direct Mapped Cache

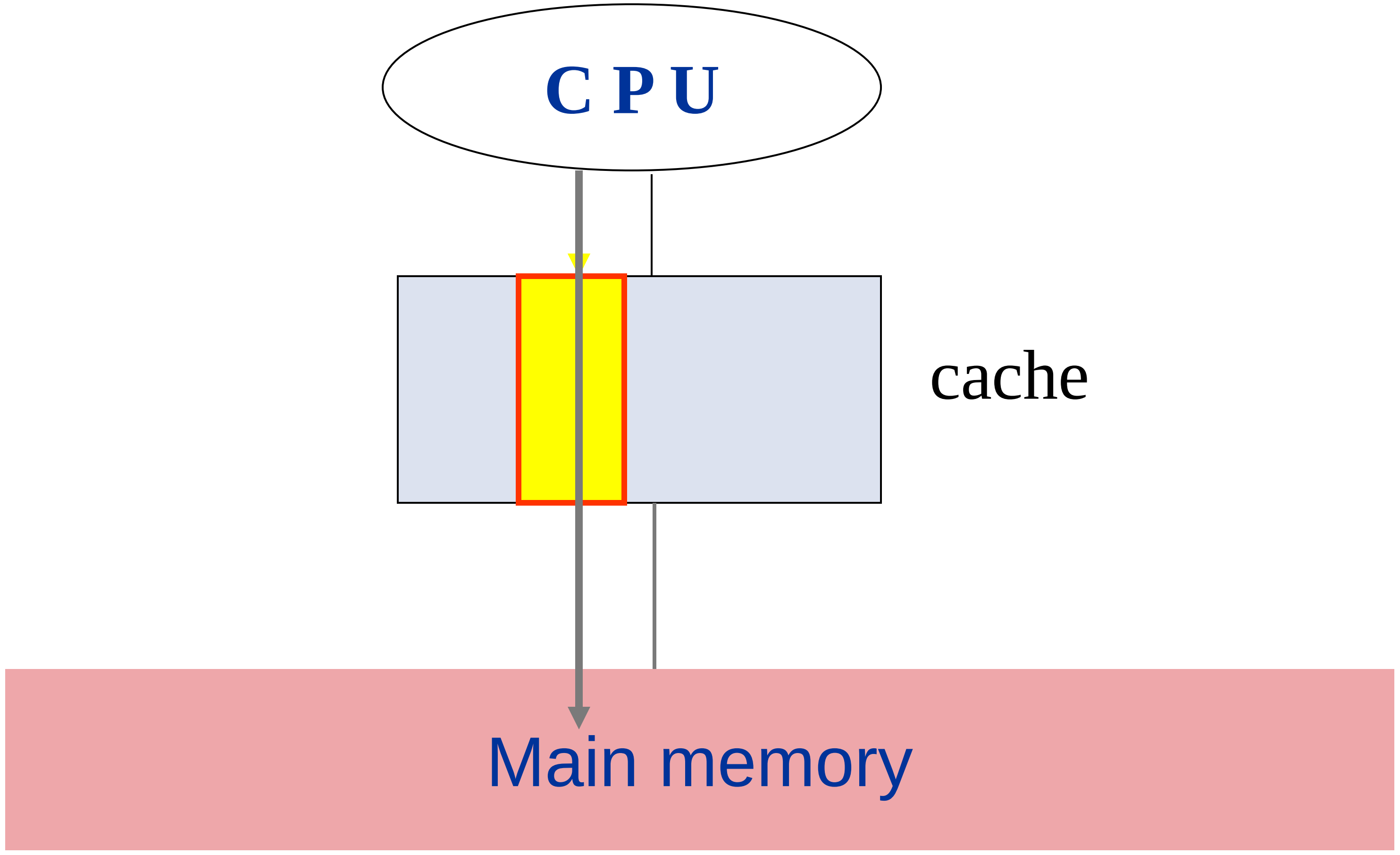
Set Associative Cache

MEMORY HIERARCHY BASICS

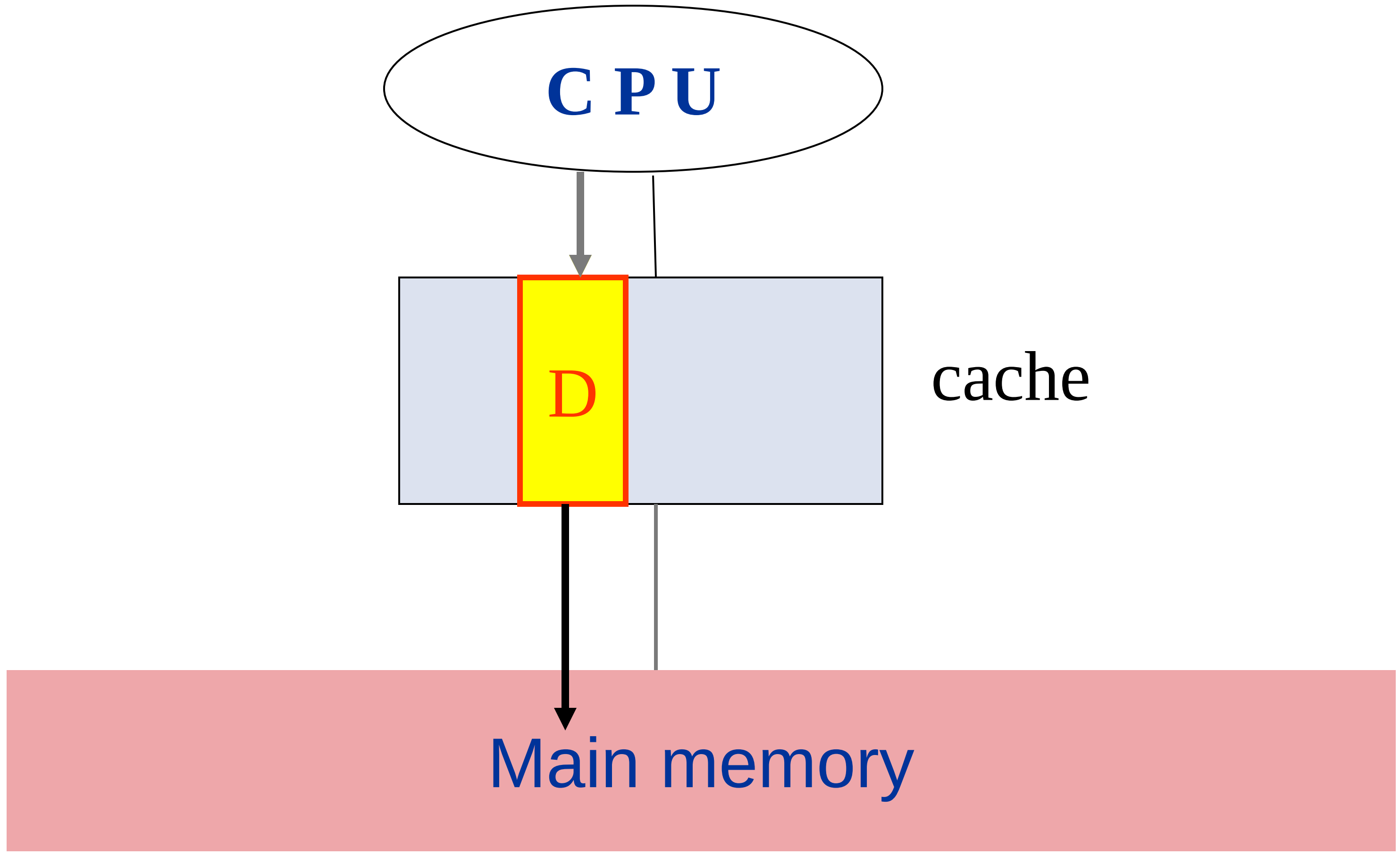
Writing to cache: two strategies

- ❑ **Write-through**
  - Immediately update lower levels of hierarchy
- ❑ **Write-back**
  - Only update lower levels of hierarchy when an updated block is replaced
- ❑ **Both strategies use write buffer to make writes asynchronous**

WRITE THROUGH



WRITE BACK



CACHE PERFORMANCE

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

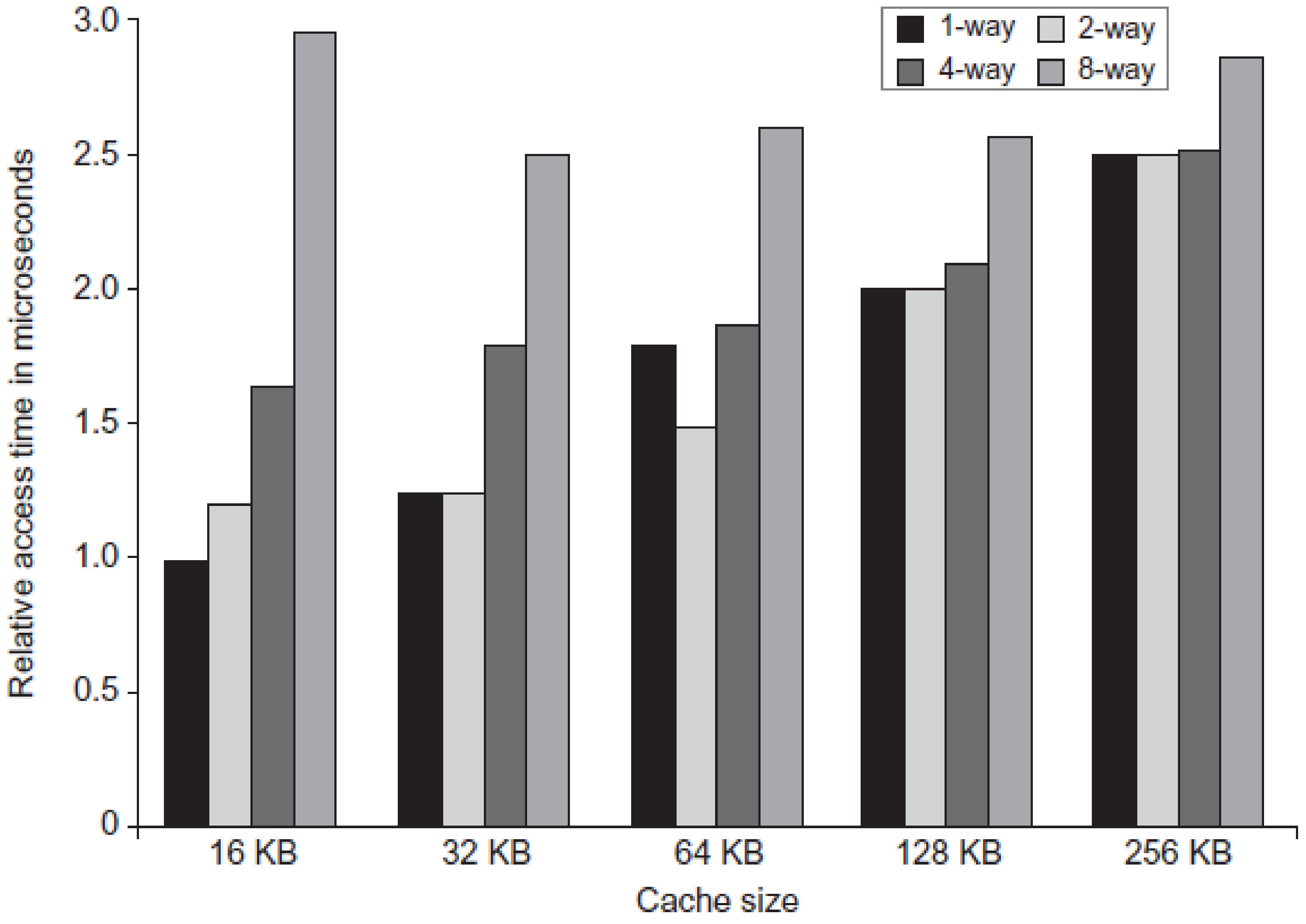
1. **Reducing the hit time**
2. **Increasing cache bandwidth**
3. **Reducing the miss penalty**
4. **Reducing the miss rate**
5. **Reducing the miss penalty or miss rate via parallelism**

OPTIMIZATIONS OF CACHE PERFORMANCE

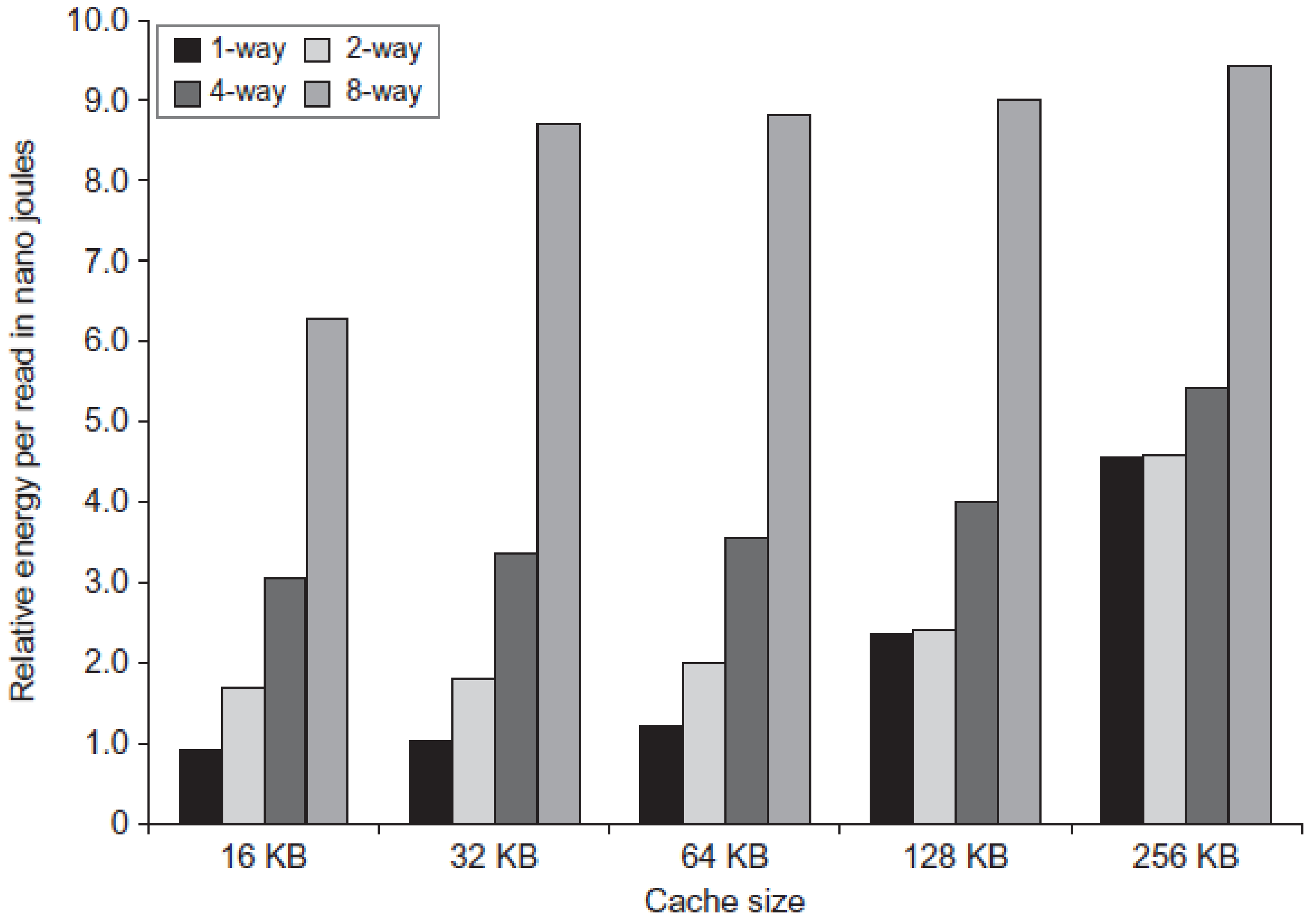
1- Small and Simple First-Level Caches to Reduce Hit Time and Power

- **Critical timing path:**
  - addressing tag memory, then
  - comparing tags, then
  - selecting correct set
- **Direct-mapped caches can overlap tag compare and transmission of data**
- **Lower associativity reduces power because fewer cache lines are accessed**

OPTIMIZATIONS OF CACHE PERFORMANCE



OPTIMIZATIONS OF CACHE PERFORMANCE



OPTIMIZATIONS OF CACHE PERFORMANCE

2- Way Prediction to Reduce Hit Time

- ❑ **To improve hit time, the multiplexor is preset to select the desired block**
  - **Prediction accuracy**
    - > 90% for two-way
    - > 80% for four-way
    - I-cache has better accuracy than D-cache

- ❑ **Block predictor bits are added to each block of a cache.**
- ❑ **The bits select which of the blocks to try on the next cache access.**

OPTIMIZATIONS OF CACHE PERFORMANCE

3- Pipelined Cache Access to Increase Bandwidth

- ❑ **Increase cache bandwidth by pipelining the cache access to allow multiple accesses per clock.**

Example from Intel:

- **Pentium: 1 cycle**
- **Pentium Pro – Pentium III: 2 cycles**
- **Pentium 4 – Core i7: 4 cycles**

OPTIMIZATIONS OF CACHE PERFORMANCE

4- Non-blocking Caches to Increase Cache Bandwidth

- For pipelined computers with (out-of-order execution), the processor need (no stall) on a data cache miss.
- A nonblocking cache or lookup-free cache allow the data cache to continue to supply cache hits during a miss.
- In general, out-of-order processors are capable of hiding much of the miss penalty of an L1 data cache miss that hits in the L2 cache but are not capable of hiding a significant fraction of a lower level cache miss.

OPTIMIZATIONS OF CACHE PERFORMANCE

5- Multi-banked Caches to Increase Cache Bandwidth

- ❑ **Increase cache bandwidth by widening the cache with multiple banks to support simultaneous access.**
  - ARM Cortex-A8 supports 1-4 banks for L2
  - Intel i7 supports 4 banks for L1 and 8 banks for L2
- ❑ **Interleave banks according to block address**

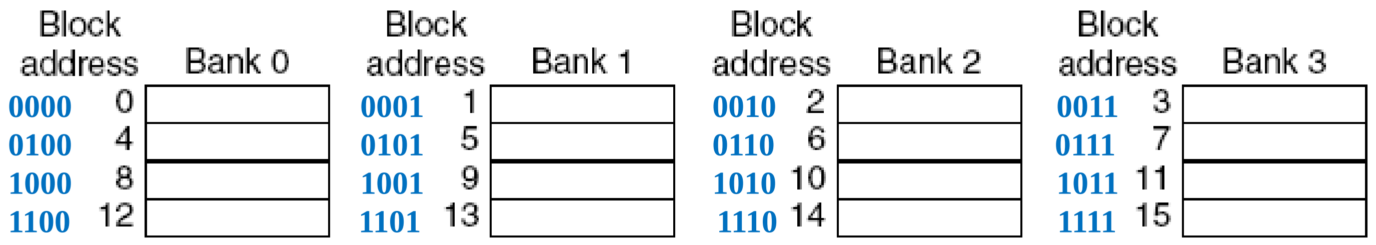


Figure 2.6 Four-way interleaved cache banks using block addressing. Assuming 64 bytes per blocks, each of these addresses would be multiplied by 64 to get byte addressing.

OPTIMIZATIONS OF CACHE PERFORMANCE

6- Critical Word First and Early Restart to Reduce Miss Penalty

❑ **Critical word first**

- Request missed word from memory first
- Send it to the processor as soon as it arrives

❑ **Early restart**

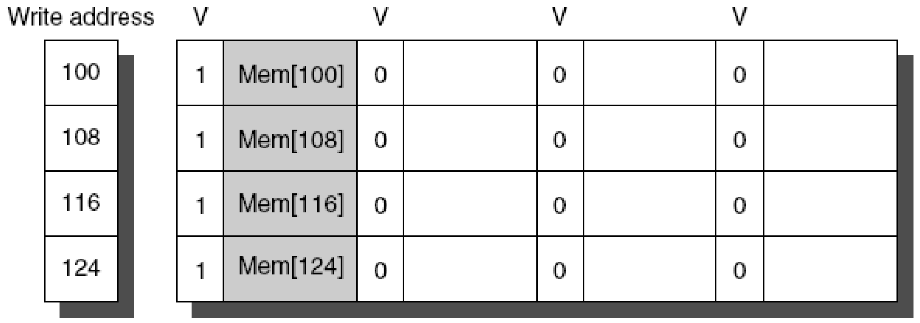
- Request words in normal order
- Send missed work to the processor as soon as it arrives

❑ **Effectiveness of these strategies depends on block size and likelihood of another access to the portion of the block that has not yet been fetched**

OPTIMIZATIONS OF CACHE PERFORMANCE

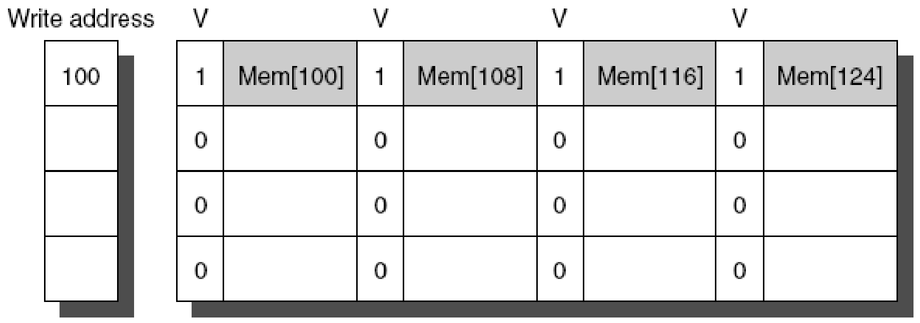
7- Merging Write Buffer to Reduce Miss Penalty

- ❑ When storing to a block that is already pending in the write buffer, update write buffer
- ❑ Reduces stalls due to full write buffer



Write address	V		V		V		V	
100	1	Mem[100]	0		0		0	
108	1	Mem[108]	0		0		0	
116	1	Mem[116]	0		0		0	
124	1	Mem[124]	0		0		0	

No write buffering



Write address	V		V		V		V	
100	1	Mem[100]	1	Mem[108]	1	Mem[116]	1	Mem[124]
	0		0		0		0	
	0		0		0		0	
	0		0		0		0	

Write buffering

OPTIMIZATIONS OF CACHE PERFORMANCE

8- Compiler Optimizations to Reduce Miss Rate

❑ **Loop Interchange**

- Swap nested loops to access memory in sequential order

❑ **Blocking**

- Instead of accessing entire rows or columns, subdivide matrices into blocks
- Requires more memory accesses but improves locality of accesses

OPTIMIZATIONS OF CACHE PERFORMANCE

9- Hardware Prefetching of Instructions and Data to Reduce Miss Penalty or Miss Rate

- **Prefetch items before the processor requests them.**
- **The processor fetches two blocks on a miss: the requested block and the next consecutive block.**
- **The requested block is placed in the instruction cache when it returns, and the prefetched block is placed in the instruction stream buffer.**
- **If the requested block is present in the instruction stream buffer, the original cache request is canceled, the block is read from the stream buffer, and the next prefetch request is issued.**

OPTIMIZATIONS OF CACHE PERFORMANCE

10- Compiler-Controlled Prefetching to Reduce Miss Penalty or Miss Rate

- **Insert prefetch instructions before data is needed**

- **Register prefetch**
  - Loads data into register
- **Cache prefetch**
  - Loads data into cache

OPTIMIZATIONS SUMMARY

Technique	Hit time	Band-width	Miss penalty	Miss rate	Power consumption	Hardware cost/ complexity	Comment
Small and simple caches	+			–	+	0	Trivial; widely used
Way-predicting caches	+				+	1	Used in Pentium 4
Pipelined cache access	–	+				1	Widely used
Nonblocking caches		+	+			3	Widely used
Banked caches		+			+	1	Used in L2 of both i7 and Cortex-A8
Critical word first and early restart			+			2	Widely used
Merging write buffer			+			1	Widely used with write through
Compiler techniques to reduce cache misses				+		0	Software is a challenge, but many compilers handle common linear algebra calculations
Hardware prefetching of instructions and data			+	+	–	2 instr., 3 data	Most provide prefetch instructions; modern high-end processors also automatically prefetch in hardware.
Compiler-controlled prefetching			+	+		3	Needs nonblocking cache; possible instruction overhead; in many CPUs

Figure 2.11 Summary of 10 advanced cache optimizations showing impact on cache performance, power consumption, and complexity. Although generally a technique helps only one factor, prefetching can reduce misses if done sufficiently early; if not, it can reduce miss penalty. + means that the technique improves the factor, – means it hurts that factor, and blank means it has no impact. The complexity measure is subjective, with 0 being the easiest and 3 being a challenge.