

CMSC 411

Computer Architecture

Lecture 20

Memory Hierarchy & Cache



Lecture's Overview

Previous Lecture:

- Pipeline hazard detection
 - Data hazard
 - Control hazard
- Controlling pipeline operations
 - Data forwarding and branch prediction
 - Coordination between control and data hazard
- Supporting exceptions
 - Pipeline flushing
 - Invoking exception handling routines

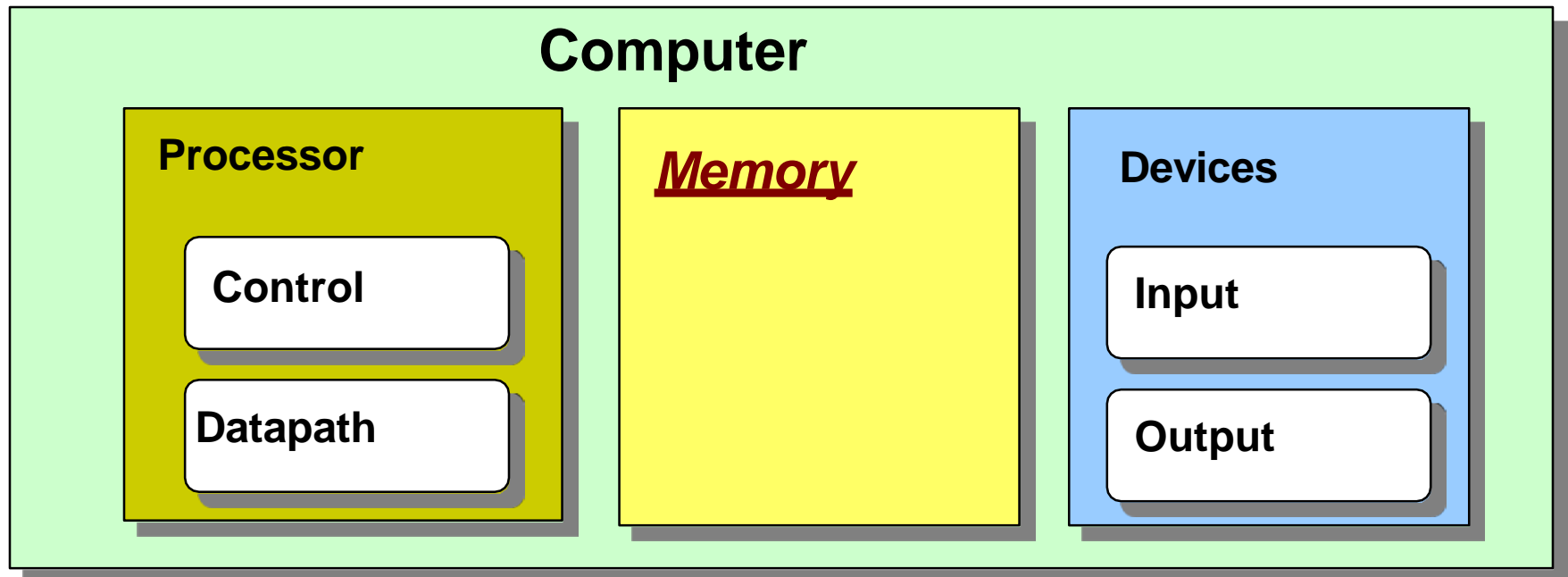
This Lecture:

- Memory hierarchy
- The basic of cache memory
- Organization of main memory

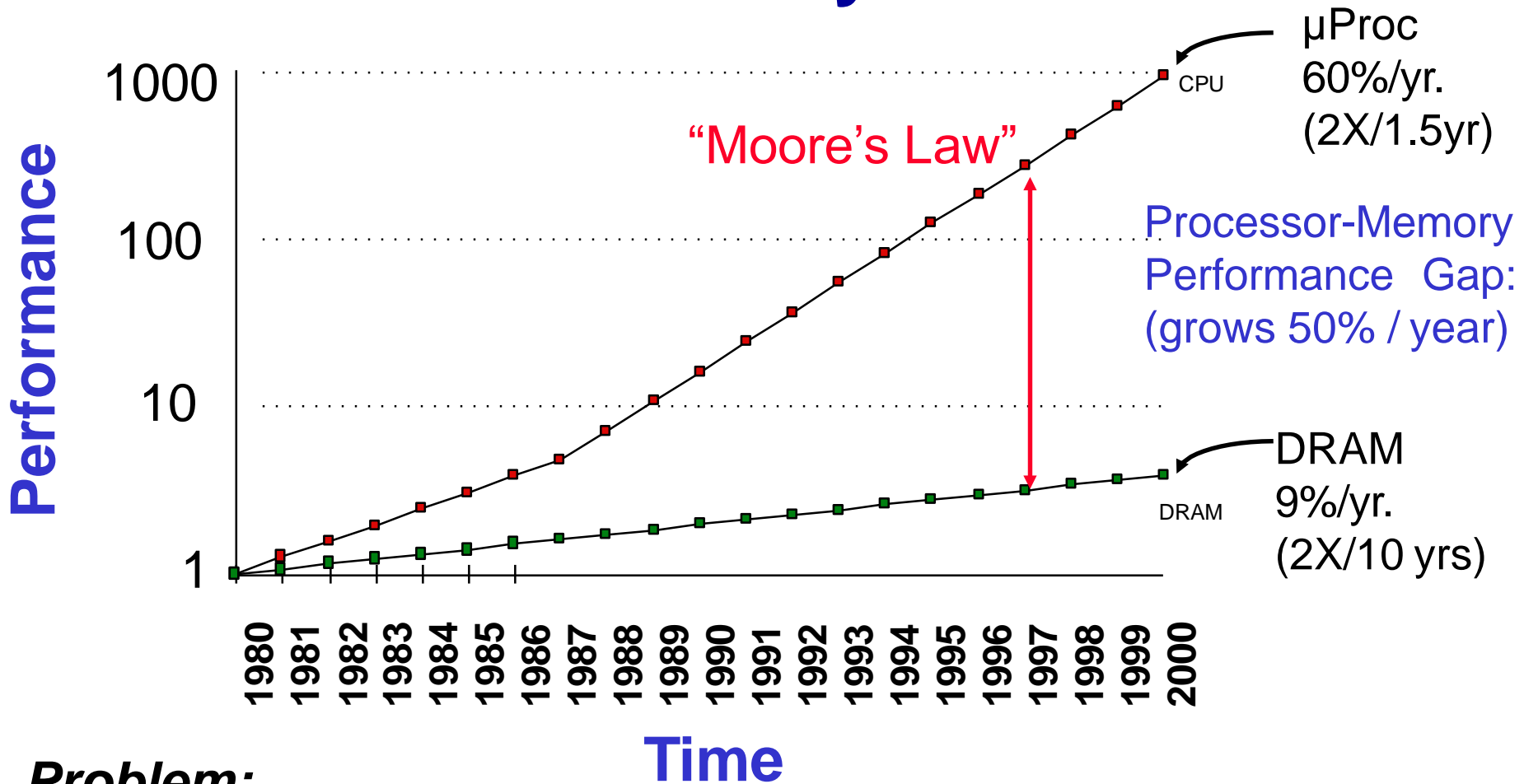


Introduction

- ❑ Why do designers need to know about Memory technology?
 - ➔ Processor performance is usually limited by memory bandwidth
 - ➔ As IC densities increase, lots of memory will fit on processor chip
- ❑ What are the different types of memory?
- ❑ Why are not memory just a bunch of flip-flops?



Processor-Memory Performance



Problem:

Memory can be a bottleneck for processor performance

Solution:

Rely on memory hierarchy of faster memory to bridge the gap



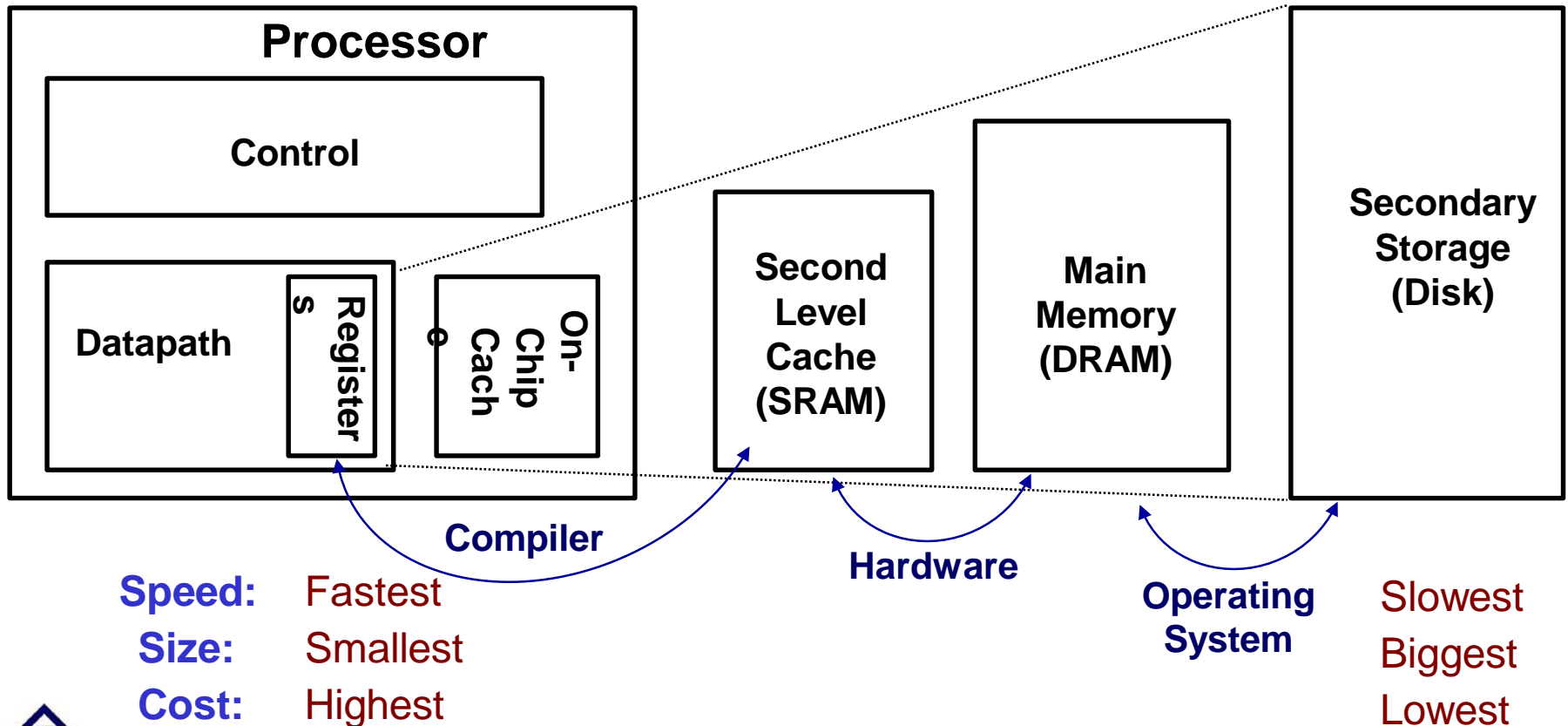
Memory Hierarchy

□ Temporal Locality (Locality in Time):

⇒ Keep most recently accessed data items closer to the processor

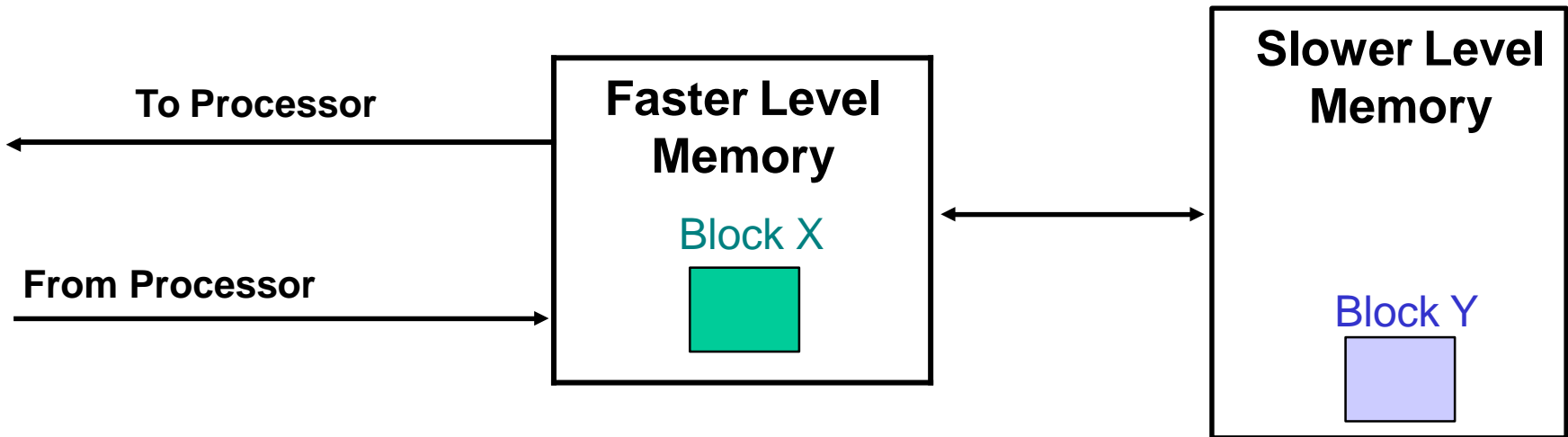
□ Spatial Locality (Locality in Space):

⇒ Move blocks of contiguous words to the faster levels



Memory Hierarchy Terminology

- **Hit:** data appear in some block in the faster level (example: Block X)
 - **Hit Rate:** the fraction of memory access found in the faster level
 - **Hit Time:** Time to access the faster level which consists of
Memory access time + Time to determine hit/miss
- **Miss:** data need to be retrieve from a block in the slower level (Block Y)
 - **Miss Rate** = 1 - (Hit Rate)
 - **Miss Penalty:** Time to replace a block in the upper level +
Time to deliver the block the processor
- **Hit Time << Miss Penalty**



* Slide is courtesy of Dave Patterson

Memory Access Types

□ Random Access:

- “Random” is good: access time is the same for all locations
- **DRAM:** Dynamic Random Access Memory
 - High density, low power, cheap, slow
 - Dynamic: need to be “refreshed” regularly
- **SRAM:** Static Random Access Memory
 - Low density, high power, expensive, fast
 - Static: content will last “forever”(until losing power)

□ “Non-so-random” Access:

- Access time varies from location to location and from time to time
- Examples: Disk, CD-ROM

□ Sequential Access:

- Access time linear in location (e.g., Tape)
- Very slow, inflexible, cheap, commonly used for backups

□ The focus of this and next lecture on random access technology

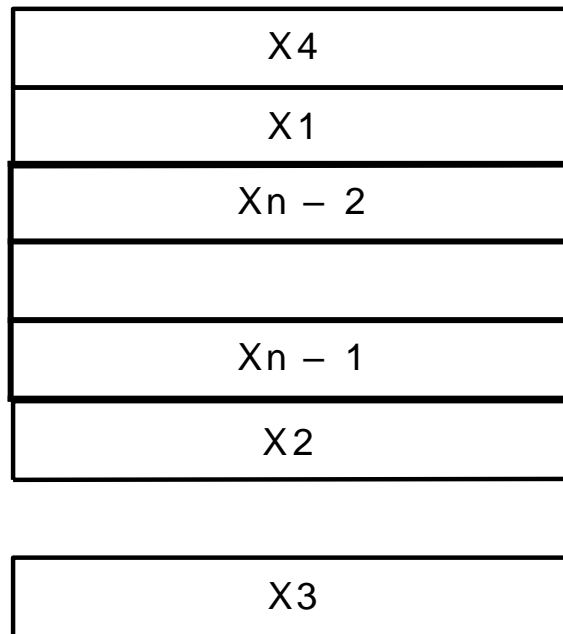
- The Main Memory: DRAMs + Caches: SRAMs



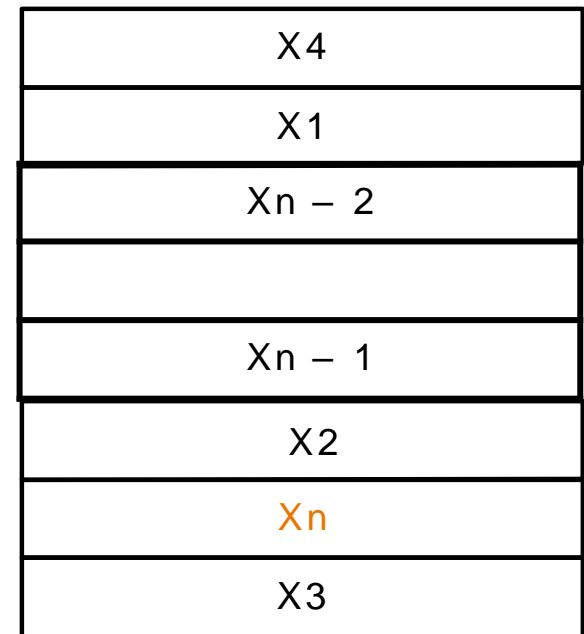
The Basics of Cache

- ❑ Cache is the name chosen to represent the first level of hierarchy between the CPU and main memory in the first commercial machine
- ❑ Caches first appeared in research machines in early 1960s
- ❑ Virtually every general-purpose computer produced today includes cache

Requesting X_n generates a miss and the word X_n will be brought from main memory to cache



a. Before the reference to X_n



b. After the reference to X_n

Issues:

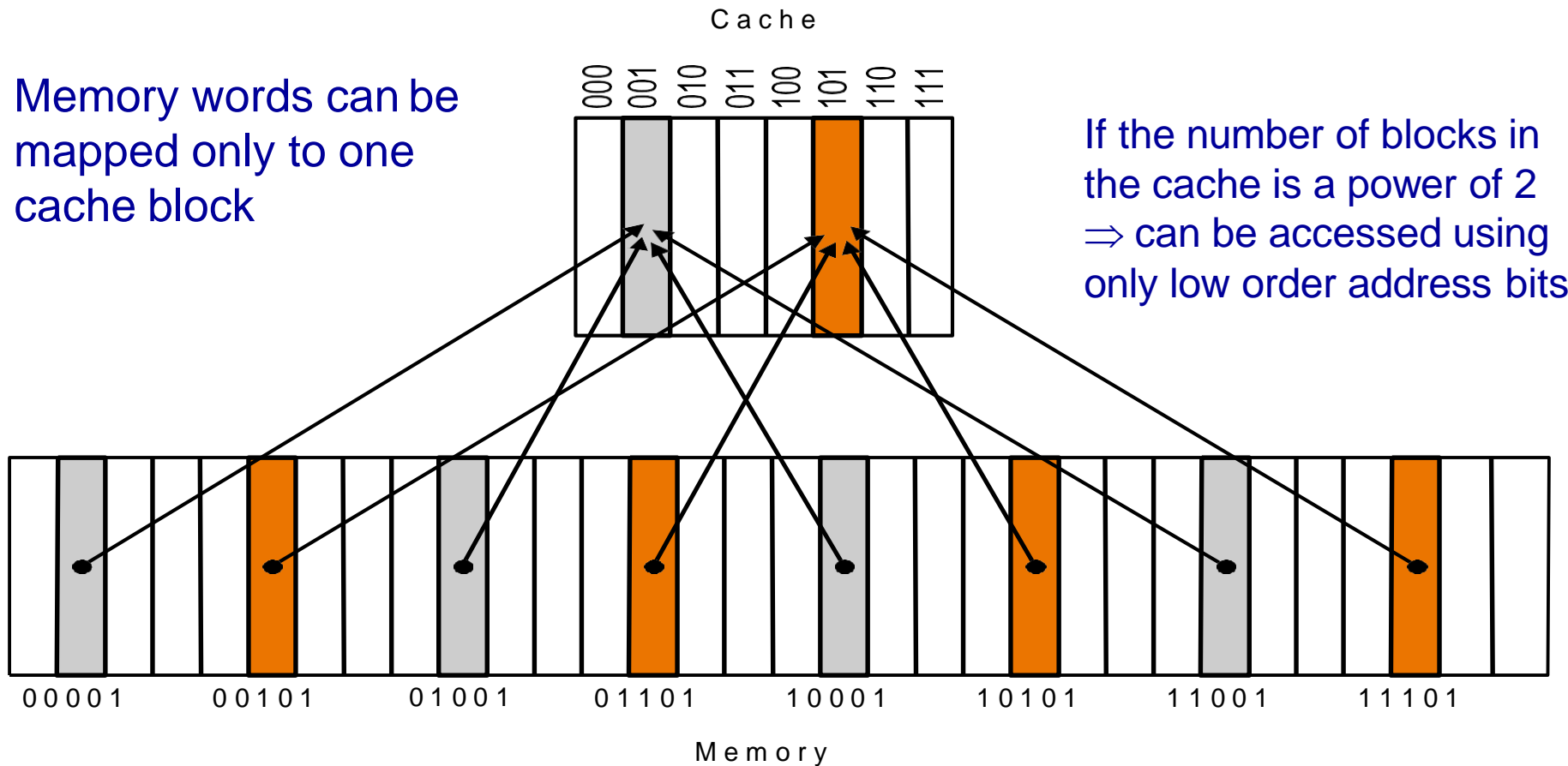
- ➔ How do we know that a data item is in cache?
- ➔ If so, How to find it?



Direct-Mapped Cache

- ❑ A tag per cache block identifies the memory address it is mapped to
- ❑ A valid bit per cache block indicates whether the content is current and active

Memory words can be mapped only to one cache block



Cache block address = (Block address) modulo (Number of cache blocks)



Accessing Cache

Cache Size depends on:

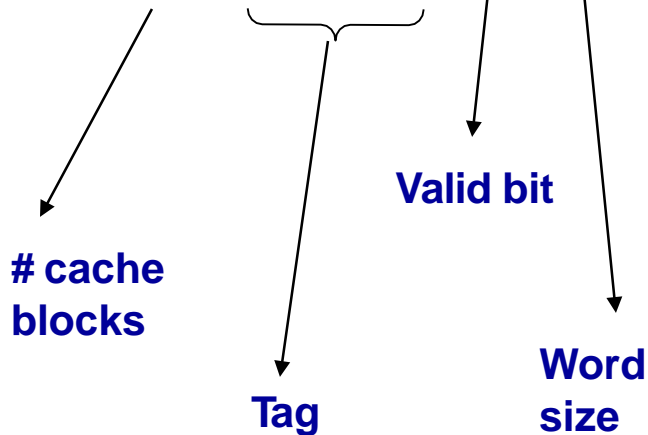
- # cache blocks
- # address bits
- Word size

Example:

For n -bit address, 4-byte word & 1024 cache blocks:

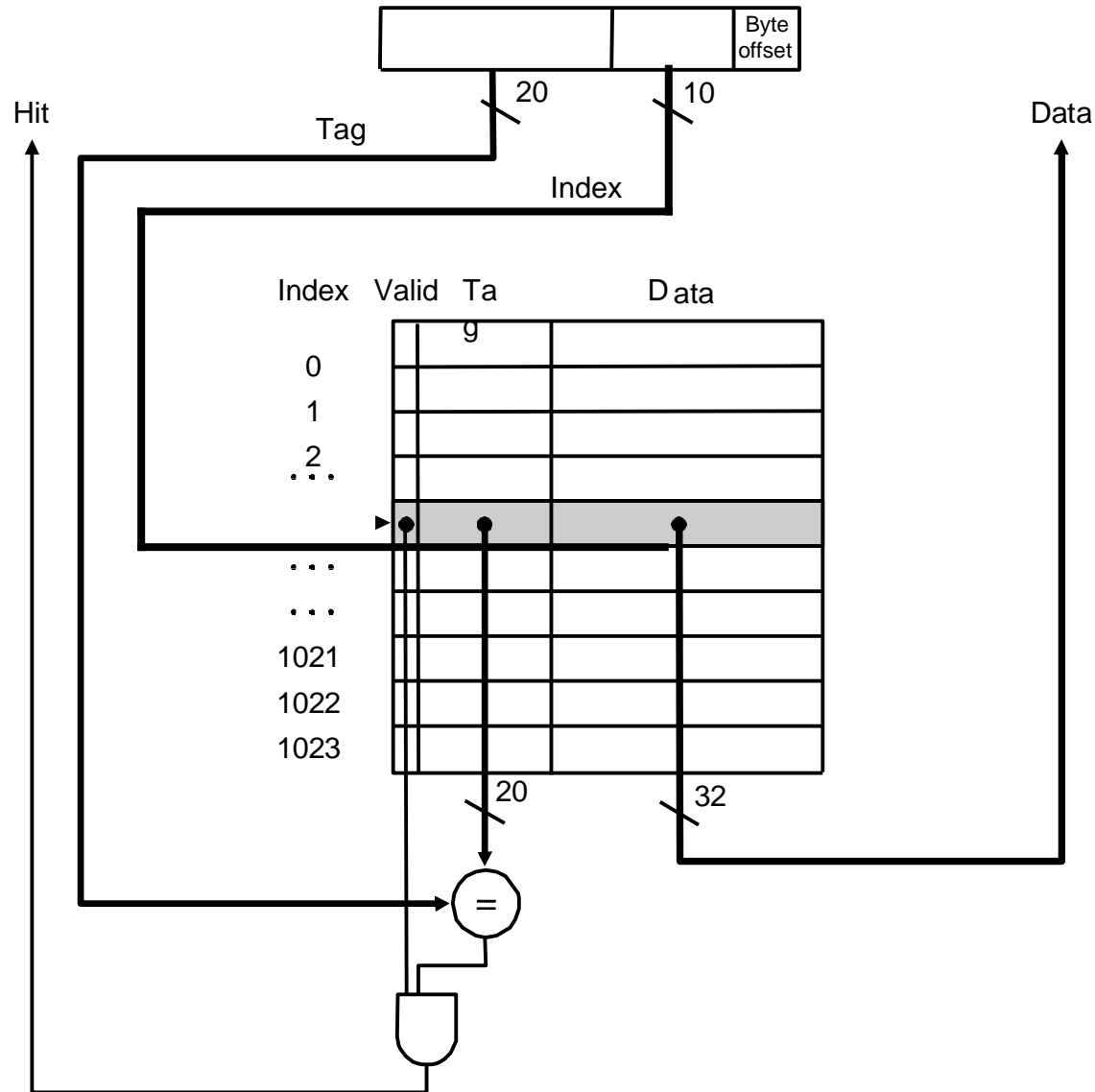
cache size =

$$1024 [(n-10) - 2] + 1 + 32 \text{ bit}$$



Address (showing bit positions)

31 30 ... 13 12 11 ... 2 1 0



Example

Reference address (Decimal)	Reference address (Binary)	Hit / miss	Assigned cache block
22	10110	Miss	$(10110 \text{ mod } 1000) = 110$
26	11010	Miss	$(11010 \text{ mod } 1000) = 010$
22	10110	Hit	$(10110 \text{ mod } 1000) = 110$
26	11010	Hit	$(11010 \text{ mod } 1000) = 010$

Index	Valid	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10	Memory(10110)
111	N		

Index	Valid	Tag	Data
000	N		
001	N		
010	Y	11	Memory(11010)
011	N		
100	N		
101	N		
110	Y	10	Memory(10110)
111	N		



Example

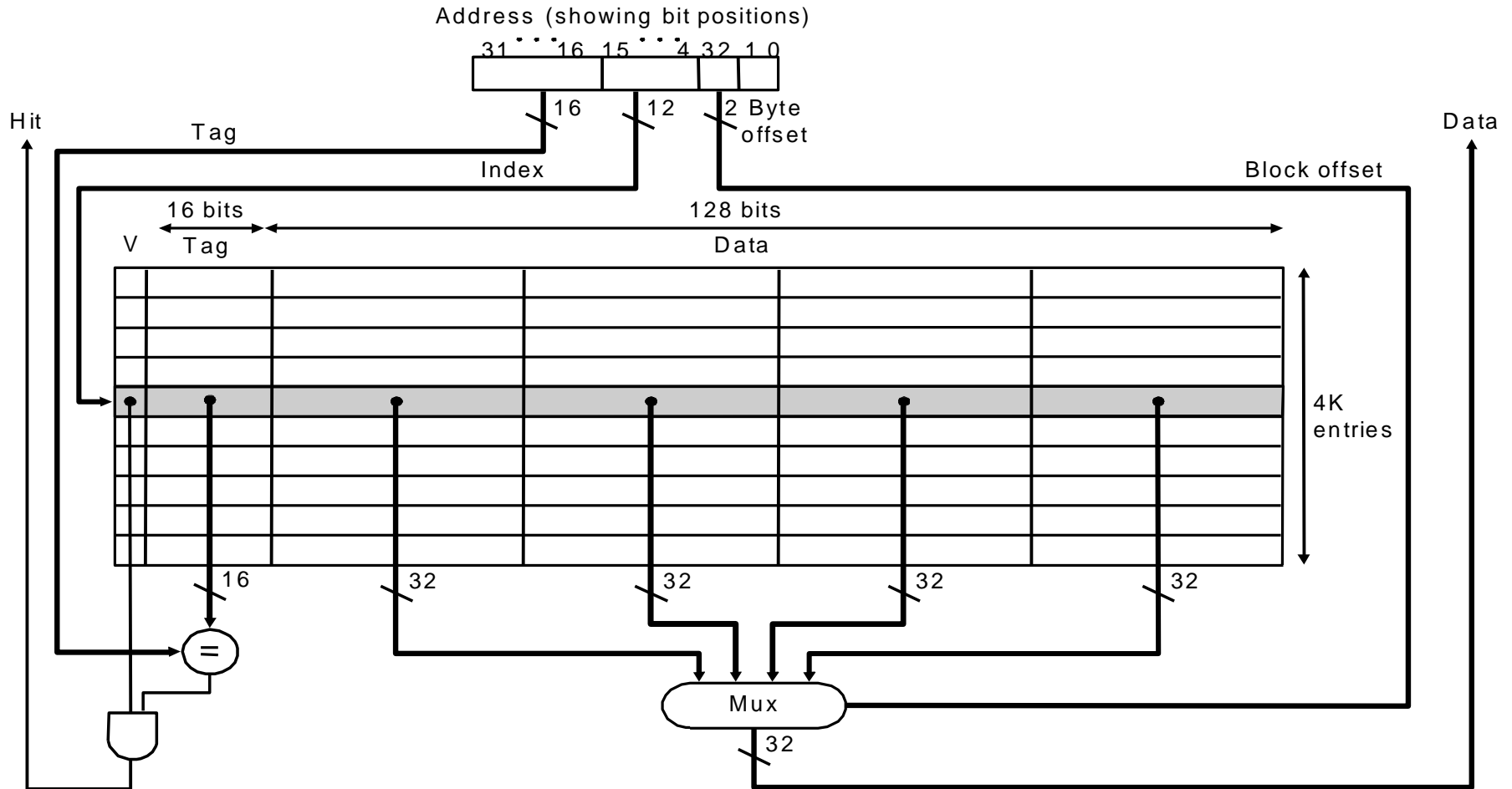
Reference address (Decimal)	Reference address (Binary)	Hit / miss	Assigned cache block
22	10110	Miss	$(10110 \text{ mod } 1000) = 110$
26	11010	Miss	$(11010 \text{ mod } 1000) = 010$
22	10110	Hit	$(10110 \text{ mod } 1000) = 110$
26	11010	Hit	$(11010 \text{ mod } 1000) = 010$
16	10000	Miss	$(10000 \text{ mod } 1000) = 000$
3	00011	Miss	$(00011 \text{ mod } 1000) = 011$
16	10000	Hit	$(10000 \text{ mod } 1000) = 000$

Index	Valid	Tag	Data
000	Y	10	Memory(10000)
001	N		
010	Y	11	Memory(11010)
011	N		
100	N		
101	N		
110	Y	10	Memory(10110)
111	N		

Index	Valid	Tag	Data
000	Y	10	Memory(10000)
001	N		
010	Y	11	Memory(11010)
011	Y	00	Memory(00011)
100	N		
101	N		
110	Y	10	Memory(10110)
111	N		



Cache with Multi-Word/Block



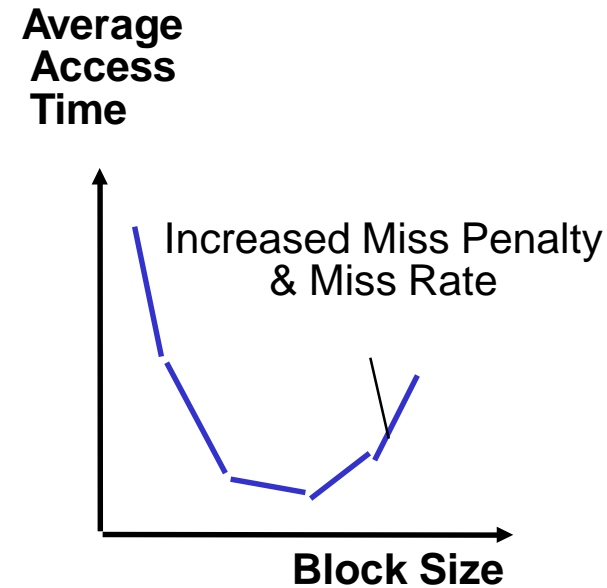
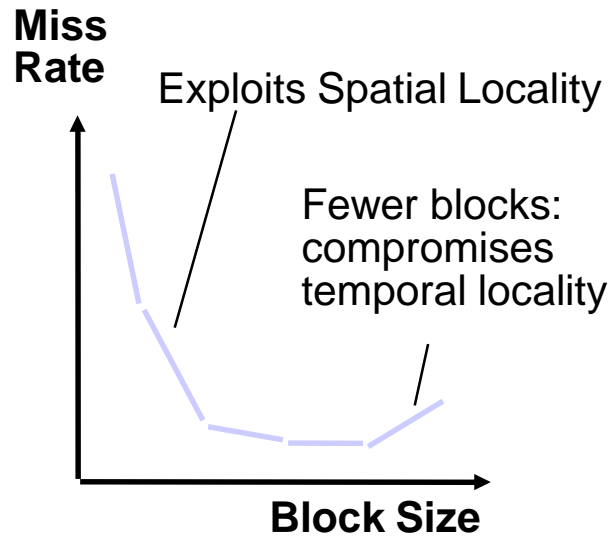
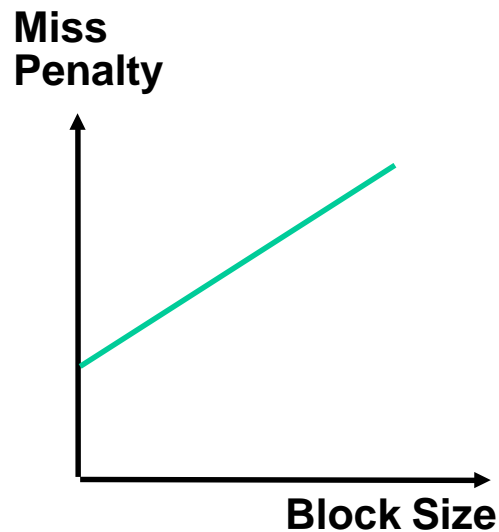
- Takes advantage of spatial locality to improve performance
- *Cache block address = (Block address) modulo (Number of cache blocks)*
- *Block address = (byte address)/(bytes per block)*



Determining Block Size

- ❑ Larger block size take advantage of spatial locality **BUT**:
 - ➔ Larger block size means larger miss penalty:
 - Takes longer time to fill up the block
 - ➔ If block size is too big relative to cache size, miss rate will go up
 - Too few cache blocks

$$\text{Average Access Time} = \text{Hit Time} \times (1 - \text{Miss Rate}) + \text{Miss Penalty} \times \text{Miss Rate}$$

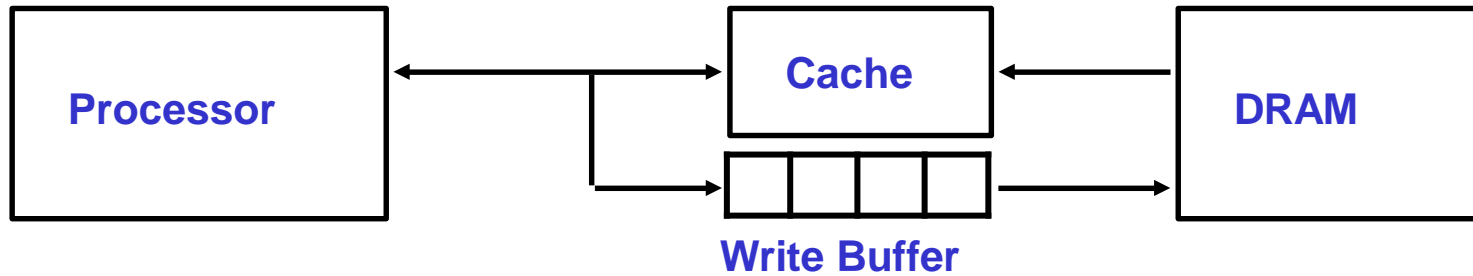


Handling Cache Misses

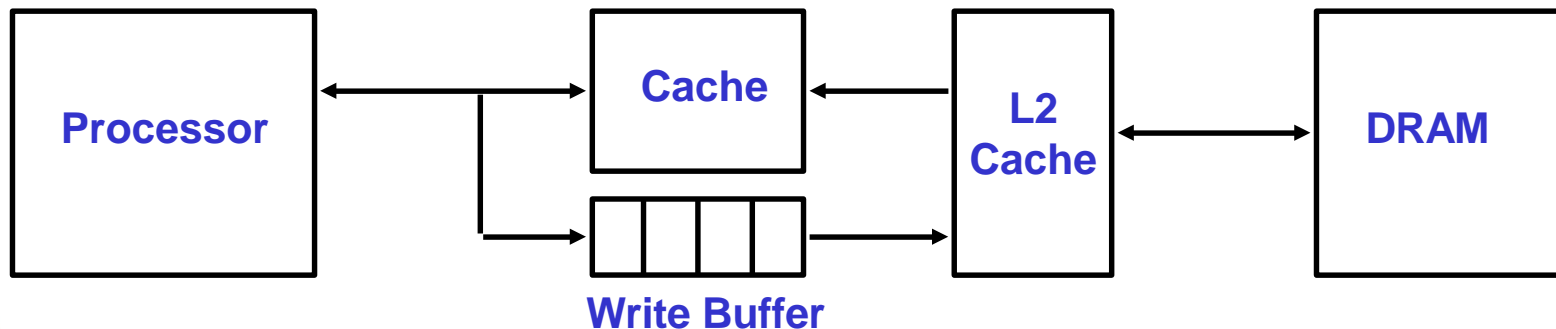
- ❑ Misses for read access always bring blocks from main memory to cache
- ❑ Write access requires careful maintenance of consistency between cache and main memory
- ❑ Two possible strategies for handling write access misses:
 - ① Write through: The information is written to both the block in the cache and to the block in the slower memory
 - ➔ Read misses cannot result in writes
 - ➔ No allocation of a cache block is needed
 - ➔ Always combined with write buffers so that don't wait for slow memory
 - ② Write back: The information is written only to the block in the cache. The modified cache block is written to main memory only when it is replaced
 - ➔ Is block clean or dirty?
 - ➔ No writes to slow memory for repeated write accesses
 - ➔ Requires allocation of a cache block



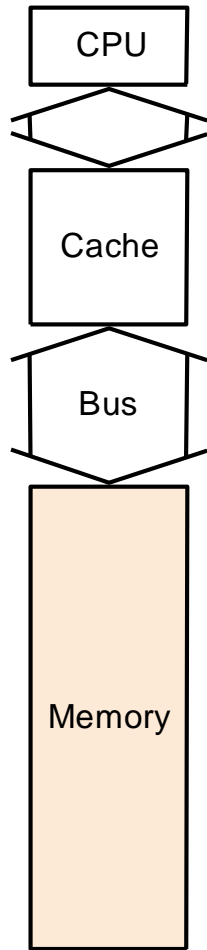
Write Through via Buffering



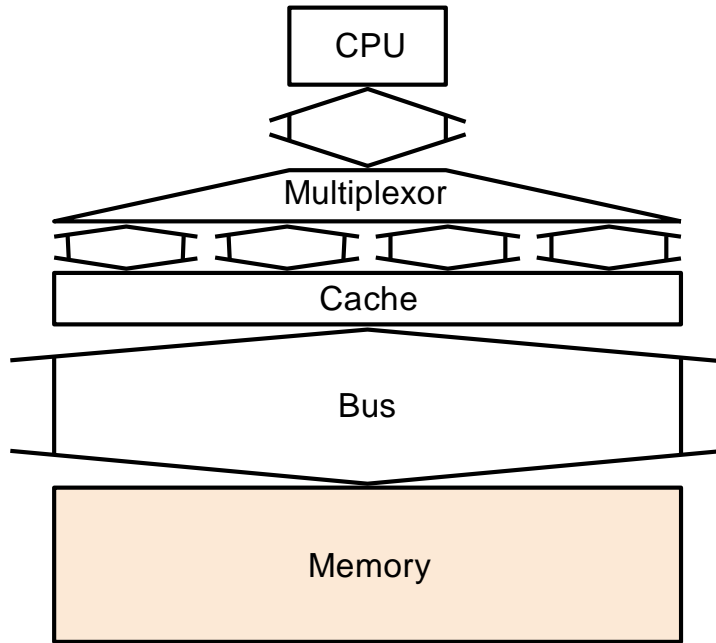
- ❑ Processor writes data into the cache and the write buffer
- ❑ Memory controller writes contents of the buffer to memory
- ❑ Increased write frequency can cause saturation of write buffer
- ❑ If CPU cycle time too fast and/or too many store instr. in a row:
 - ➔ Store buffer will overflow no matter how big you make it
 - ➔ The CPU Cycle Time get closer to DRAM Write Cycle Time
- ❑ Write buffer saturation can be handled by installing a second level (L2) cache



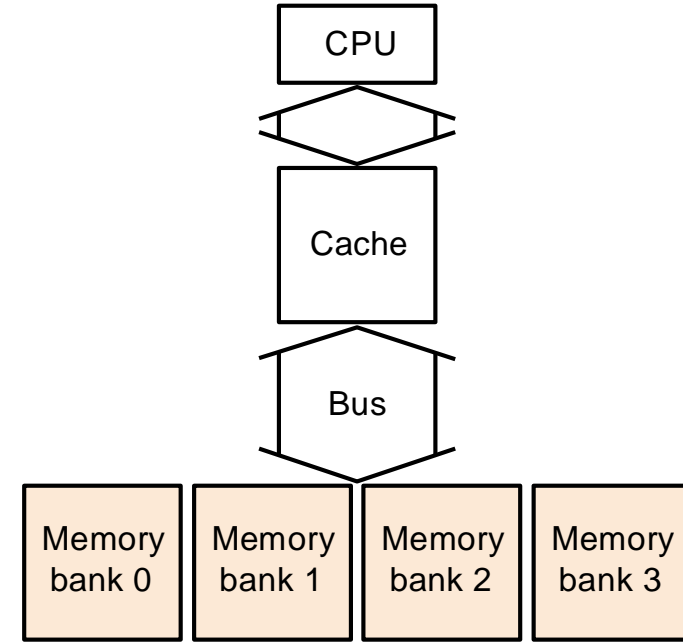
Memory Organization



a. One-word-wide memory organization



b. Wide memory organization



c. Interleaved memory organization

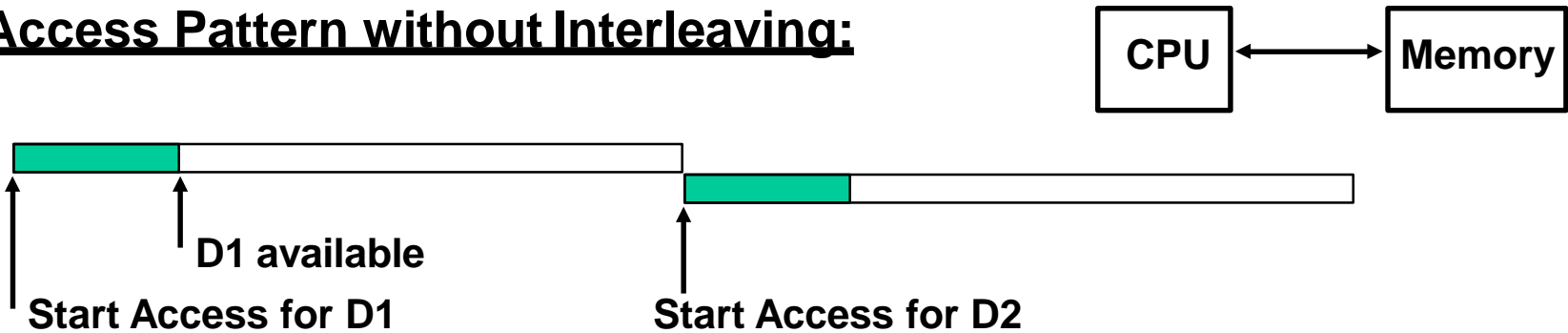
- ❑ *Simple*: CPU, Cache, Bus, Memory same width (32 bits)
- ❑ *Wide*: CPU/Mux 1 word; Mux/Cache, Bus, Memory N words
- ❑ *Interleaved*: CPU, Cache, Bus 1 word: Memory N Modules (4 Modules); example is *word interleaved*

Memory organization would have significant effect on bandwidth

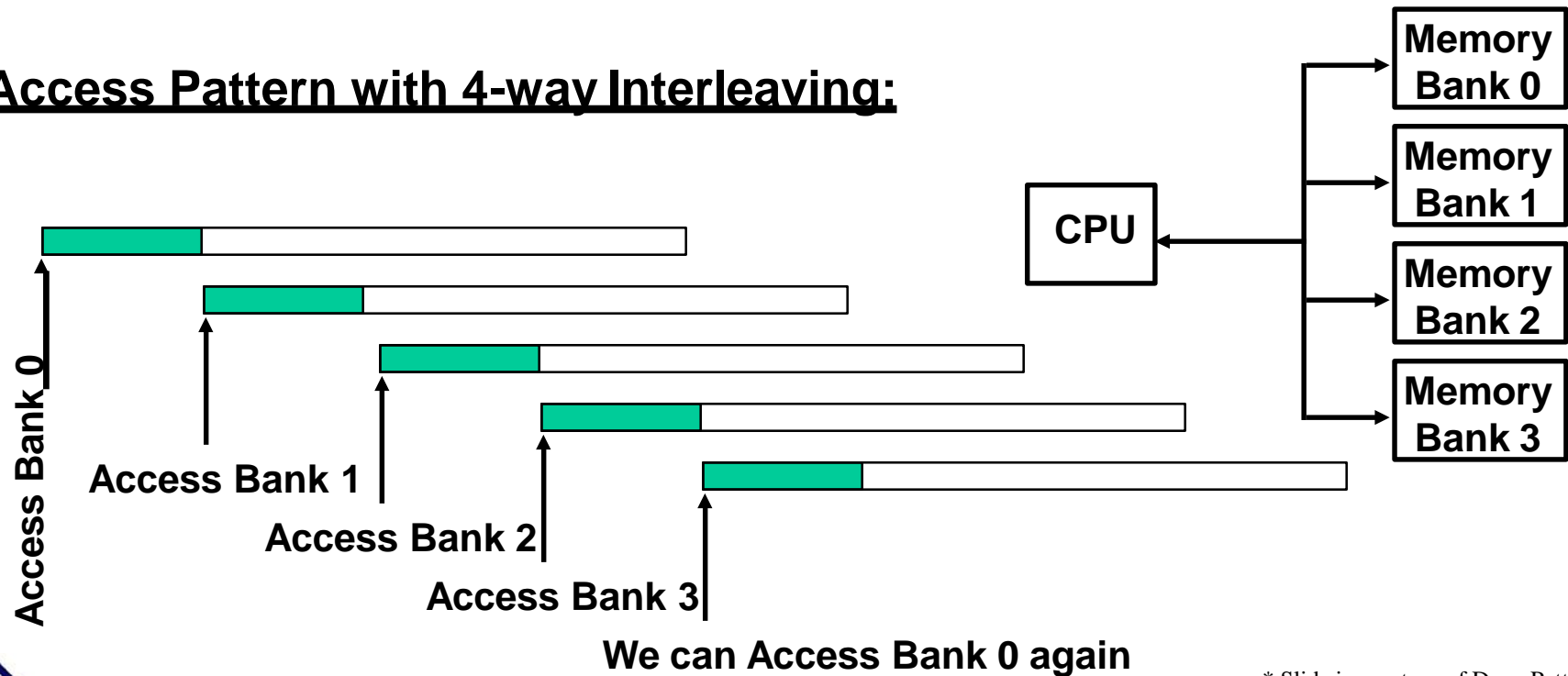


Memory Interleaving

Access Pattern without Interleaving:



Access Pattern with 4-way Interleaving:



Conclusion

□ Summary

→ Memory hierarchy

- Principles of locality
- Types of memory

→ The basics of cache memory

- Direct-mapped cache
- Handling of cache misses
- Consistency between cache and main memory

→ Memory organization

- Main memory performance issues
- Memory interleaving

□ Next Lecture

→ Measuring and improving cache performance

Read sections 5.1 & 5.2 in 5th Ed. of the textbook

