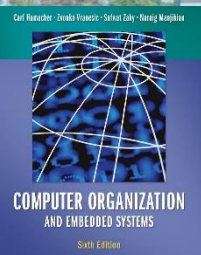香港中文大學
The Chinese University of Hong Kong

CSCI2510 Computer Organization
Lecture 08: Cache Performance

Ming-Chang YANG

mcyang@cse.cuhk.edu.hk

Reading: Chap. 8.7

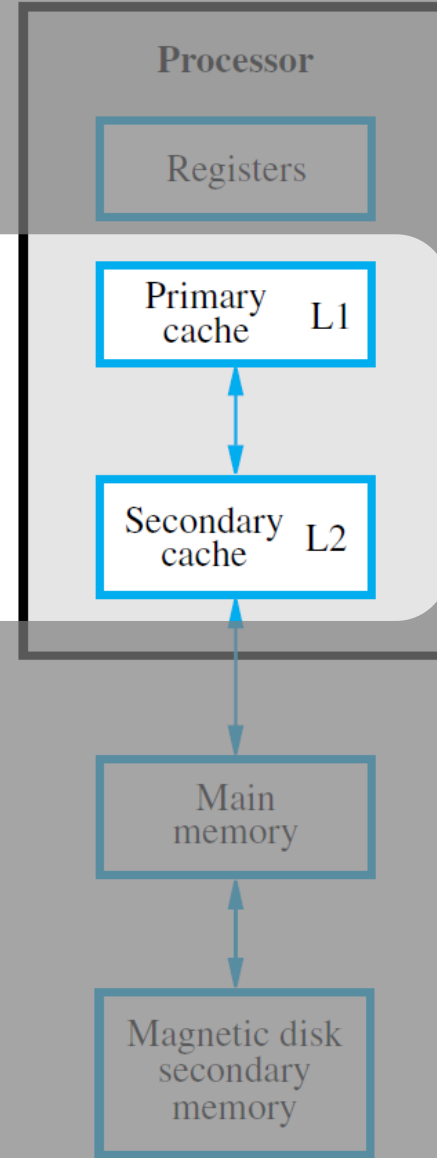# Recall: Memory Hierarchy

Processor

- Register: SRAM

- L1, L2 cache: SRAM

- Main memory: SDRAM

- Secondary storage:
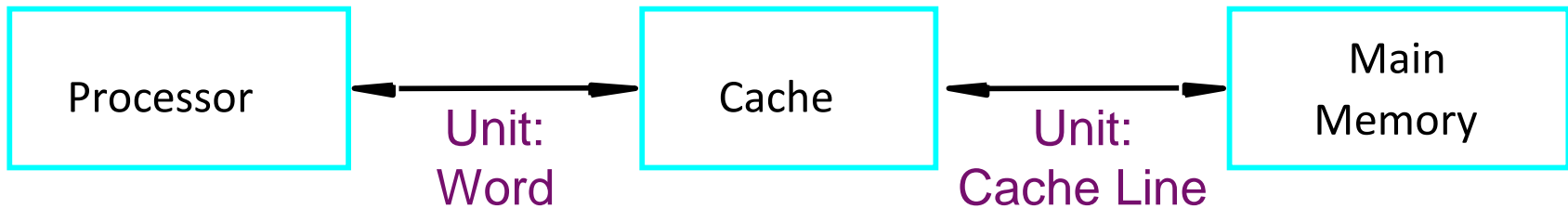 Hard disks or NVM

Increasing size

Increasing speed

Increasing cost per bit

Processor

Registers

Primary cache   L1

Secondary cache   L2

Main memory

Magnetic disk secondary memory

Recall: Cache Usage

- **Cache Read (or Write) Hit/Miss**: The read (or write) operation can/cannot be performed on the cache.

Processor	←→ Unit: Word	Cache	←→ Unit: Cache Line	Main Memory

- **Cache Block / Line**: The unit composed of multiple successive memory words (size: cache block > word).
  - The contents of a cache block (of memory words) will be loaded into or unloaded from the cache at a time.

- **Mapping Functions**: Decide how cache is organized and how addresses are mapped to the main memory.

- **Replacement Algorithms**: Decide which item to be unloaded from cache when cache is full.

Outline

- Performance Evaluation
  - Cache Hit/Miss Rate and Miss Penalty
  - Average Memory Access Time

- Performance Enhancements (Optional)
  - Prefetch
  - Memory Module Interleaving
  - Load-Through
  - Write Buffer

Performance Considerations

- Two key factors are performance and cost.
  - **Performance**: how fast instructions can be brought into the processor and how fast they can be executed.
- **Price–Performance Ratio**: the lower, the better.
  - Widely known as cost–performance ratio.
- The objective of the memory hierarchy is to look for the best price–performance ratio.
  - The processor is able to access instructions and data more quickly if the requested data are in the small cache.
  - The impact of a cache on performance is dependent on the cache hit/miss rates.
    - That is, how frequently the requested instructions and data can be found in the cache.

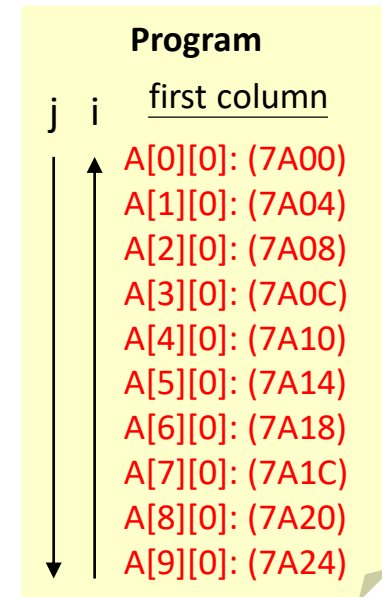# Cache Hit/Miss Rate and Miss Penalty

- A successful access to data in a cache is a hit.
  - **Hit Rate**: The ratio of number of hits to all accesses.
  - **Miss Rate**: The ratio of number of misses to all accesses.
- **Cache Hit**:
  - The access can be done in the cache.
  - Hit rates over 0.9 are essential for high-performance PCs.
- **Cache Miss**:
  - When cache miss occur, extra time is needed to bring blocks from the slower main memory to the faster cache.
  - During that time, the processor is stalled.
    - The waiting time depends on the cache operation details.
  - **Miss Penalty**: the total access time passed through (seen by the processor) when a cache miss occurs.

- In the working examples discussed in Lecture 07, the program will lead to
  - **2** cache hits (when i = 9 and 8) for direct mapping.
  - **8** cache hits (when i = 9, 8, …, 2) for associative mapping.
  - **4** cache hits (when i = 9, 8, …, 6) for 4-way set associative.
- What are the cache hit and miss rates for each case?

Program

j i   first column

A[0][0]: (7A00)
A[1][0]: (7A04)
A[2][0]: (7A08)
A[3][0]: (7A0C)
A[4][0]: (7A10)
A[5][0]: (7A14)
A[6][0]: (7A18)
A[7][0]: (7A1C)
A[8][0]: (7A20)
A[9][0]: (7A24)

An Example of Miss Penalty

- **Miss Penalty**: the total access time passed through (seen by the processor) when a cache miss occurs.

- Consider a system with only one level of cache with following parameters:

  CPU $\xleftrightarrow{t}$ Cache $\xleftrightarrow[t]{10t}$ Main Memory
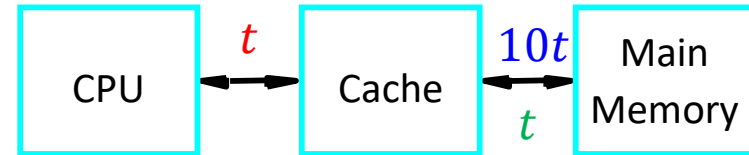
  – Word access time to the **cache**: t
  – Word access time to the **main memory**: $10t$
  – When a cache miss occurs, a cache block of <u>8 words</u> will be transferred from the main memory to the cache.
    - Time to transfer the **first word**: $10t$
    - Time to transfer **one word of the rest 7 words**: t (hardware support!)

- The miss penalty can be derived as:
$$t + 10t + 7 \times t + t = 19t$$

The initial cache access that results in a miss.

CPU access the requested data in the cache.

Outline

- ## Performance Evaluation
  - – Cache Hit/Miss Rate and Miss Penalty
  - – Average Memory Access Time

- ## Performance Enhancements (Optional)
  - – Prefetch
  - – Memory Module Interleaving
  - – Load-Through
  - – Write Buffer

Average Memory Access Time

- Consider a system with only one level of cache:
  - h: Cache Hit Rate
  - $1 - h$: Cache Miss Rate
  - C: Cache Access Time
  - M: Miss Penalty
    - It mainly consists of the time to access a block in the main memory.

- The average memory access time can be defined as:

$$t_{avg} = h \times C + (1 - h) \times M$$
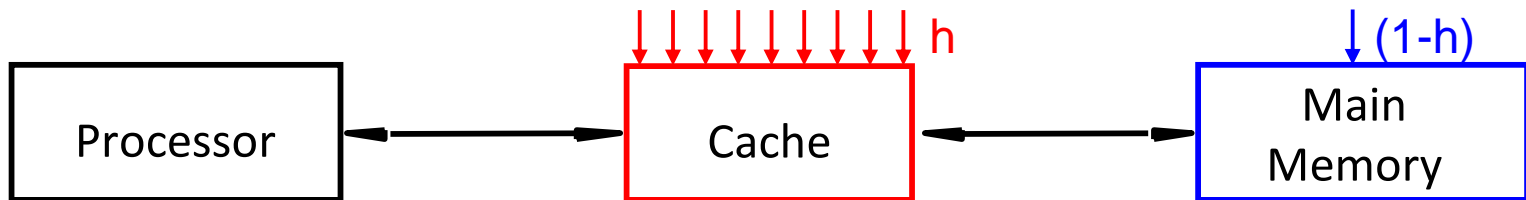
  - For example, given $h = 0.9, \ C = 1 \ cycle, \ M = 19 \ cycles$:

  → Avg. memory access time: $0.9 \times 1 + 0.1 \times 19 = 2.8 \ (cycles)$

Performance Improved by Cache (1/2)

- Consider the system with only one level of cache:
  - Word access time to the cache: 1 cycle
  - Word access time to the main memory: 10 $cycles$
  - When a cache miss occurs, a cache block of <u>8 words</u> will be transferred from the main memory to the cache.
    - Time to transfer the first word: 10 $cycles$
    - Time to transfer one word of the rest 7 words: 1 cycle
  - Miss Penalty: $1 + 10 + 7 \times 1 + 1 = 19\ (cycles)$

- Assume there are total <u>130</u> memory accesses:
  - <u>100</u> memory accesses for instructions with hit rate 0.95
  - <u>30</u> memory access for data (operands) with hit rate = 0.90

- Total execution cycles **without** cache:

$$t_{without} = 100 \times 10 + 30 \times 10 = 1300 \; cycles$$

  - All of the memory accesses will result in a reading of a memory word (of latency 10 cycles).

- Total execution cycles **with** cache:

Avg. memory access time for instructions: $h \times C + (1-h) \times M$

$$t_{with} = 100 \times (0.95 \times 1 + 0.05 \times 19) +$$
$$30 \times (0.9 \times 1 + 0.1 \times 19) = 274 \; cycles$$

Avg. memory access time for data: $h \times C + (1-h) \times M$

- The performance improvement:

$$\frac{t_{without}}{t_{with}} = \frac{1300}{274} = 4.74 \; (speed \; up!)$$

- Consider the same system with one level of cache.
  - Word access time to the cache: 1 cycle
  - Word access time to the main memory: 10 $cycles$
  - Miss Penalty: $1 + 10 + 7 \times 1 + 1 = 19 \ (cycles)$
- What is the performance difference between this cache and an ideal cache?
 - **Ideal Cache**: All the accesses can be done in cache.

Real-life Example: Intel Core 2 Duo

- Number of Processors : 1
- Number of Cores : **2** per processor
- Number of Threads : 2 per processor
- Name : **Intel Core 2 Duo E6600**
- Code Name : Conroe
- Specification : Intel(R) Core(TM)2 CPU 6600@2.40GHz
- Technology : 65 nm
- Core Speed : 2400 MHz
- Multiplier x Bus speed : 9.0 x 266.0 MHz = 2400 MHz
- Front-Side-Bus speed : 4 x 266.0MHz = 1066 MHz
- Instruction Sets : MMX, SSE, SSE2, SSE3, SSSE3, EM64T
- **L1 Cache**
  - **Data Cache** : **2 x** 32 KBytes, 8-way set associative, 64-byte line size
  - **Instruction Cache** : **2 x** 32 KBytes, 8-way set associative, 64-byte line size
- **L2 Cache** : 4096 KBytes, 16-way set associative, 64-byte line size

Real-life Example: Intel Core 2 Duo
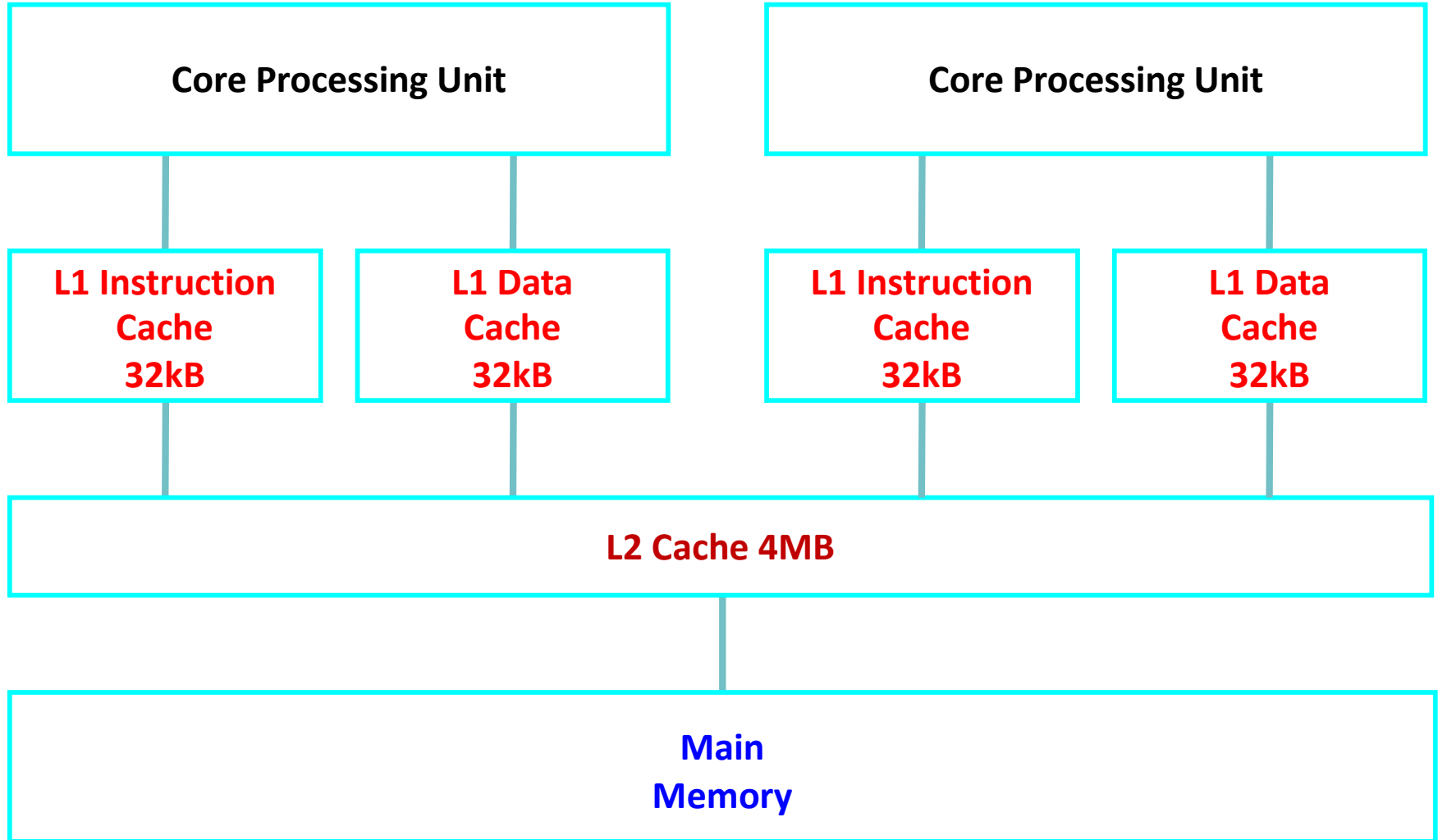
Cache on Real-life Processor

- In high-performance processors, two levels of caches are normally used, L1 and L2.
  - **L1 Cache**: Must be very fast as they determine the memory access time seen by the processor.
  - **L2 Cache**: Can be slower, but it should be much larger than the L1 cache to ensure a high hit rate.

- The avg. memory access time of two levels of caches:

$$t_{avg} = h_1 \times C_1 + (1 - h_1) \times [h_2 \times C_2 + (1 - h_2) \times M],$$

- h_1 : L1 cache hit rate
- h_2 : L2 cache hit rate
- C_1 : access time to L1 cache
- C_2 : time to transfer data from L2 to L1
- M : miss penalty to transfer data from memory → L2 → L1

The avg. memory access time of one level of cache:

$$t_{avg} = h \times C + (1 - h) \times M$$

- Given a system with one level of cache, and a system with two level of caches.

- Assume the hit rates of L1 cache and L2 cache (if any) are both 0.9.

- What are the probabilities that miss penalty has to be paid to read a block from memory in both systems?

Outline

- Performance Evaluation
  - Cache Hit/Miss Rate and Miss Penalty
  - Average Memory Access Time

- Performance Enhancements (Optional)
  - Prefetch
  - Memory Module Interleaving
  - Load-Through
  - Write Buffer

How to Improve the Performance?

- Recall the system with only one level of cache:
  - h: Cache Hit Rate
  - $1 - h$: Cache Miss Rate
  - C: Cache Access Time
  - M: Miss Penalty
    - It mainly consists of the time to access a block in the main memory.
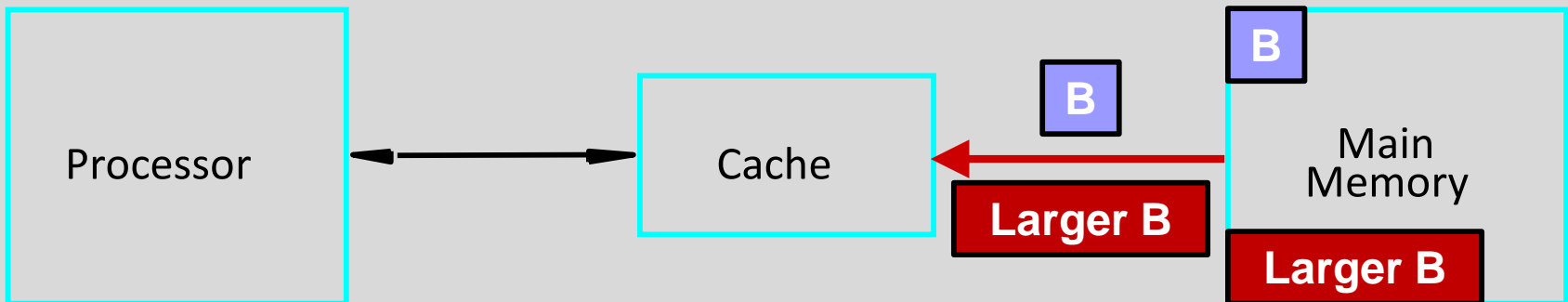
- The average memory access time can be defined as:

$$t_{avg} = h \times C + (1 - h) \times M$$

- Possible ways to further reduce t_{avg}:
  - Use faster cache (i.e., $C \uparrow$)? $$$...
  - Improve the hit rate (i.e., $h \uparrow$)?
  - Reduce the miss penalty (i.e., $M \downarrow$)?

How to Improve Hit Rate?

- How about larger block size?

  - Larger blocks take more advantage of the spatial locality.

    - **Spatial Locality**: If all items in a larger block are needed in a computation, it is better to load them into cache in a single miss.

  - Larger blocks are effective only up to a certain size:

    - Too many items will remain unused before the block is replaced.
    - It takes longer time to transfer larger blocks, and may also increase the miss penalty.

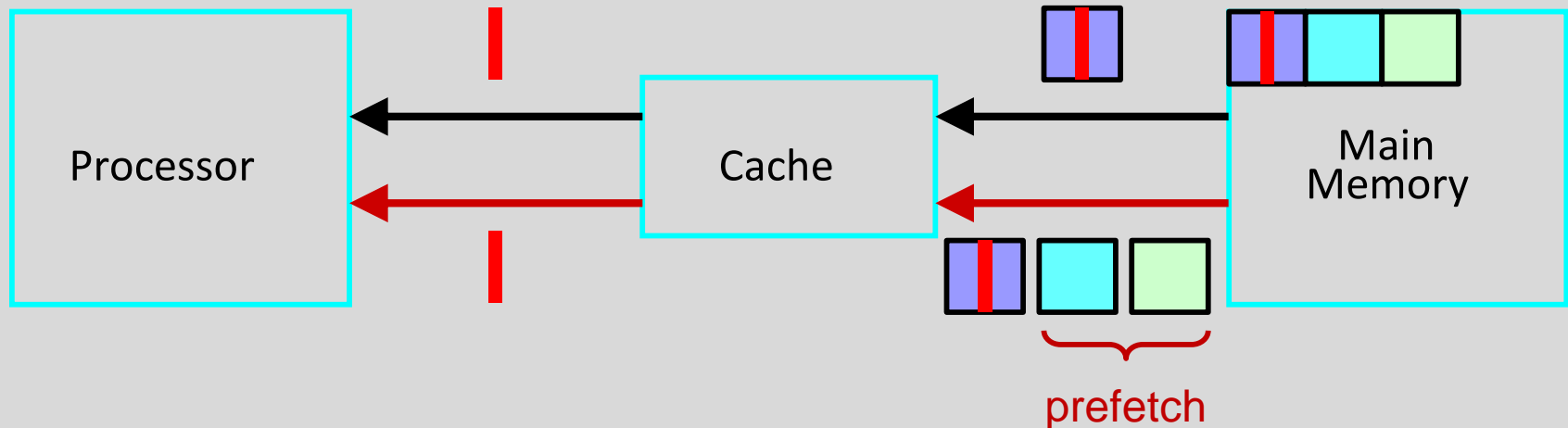 - Block sizes of 16 to 128 bytes are most popular.

Outline

- Performance Evaluation
  - Cache Hit/Miss Rate and Miss Penalty
  - Average Memory Access Time

- Performance Enhancements (Optional)
  - Prefetch
  - Memory Module Interleaving
  - Load-Through
  - Write Buffer

Prefetch: More rather than Larger

- **Prefetch**: Load more (rather than larger) blocks into the cache before they are needed, while CPU is busy.
  - Prefetch instruction can be put by programmer or compiler.

- Some data may be loaded into the cache without being used, before the prefetched data are replaced.
  - The overall effect on performance is positive.
  - Most processors support the prefetch instruction.
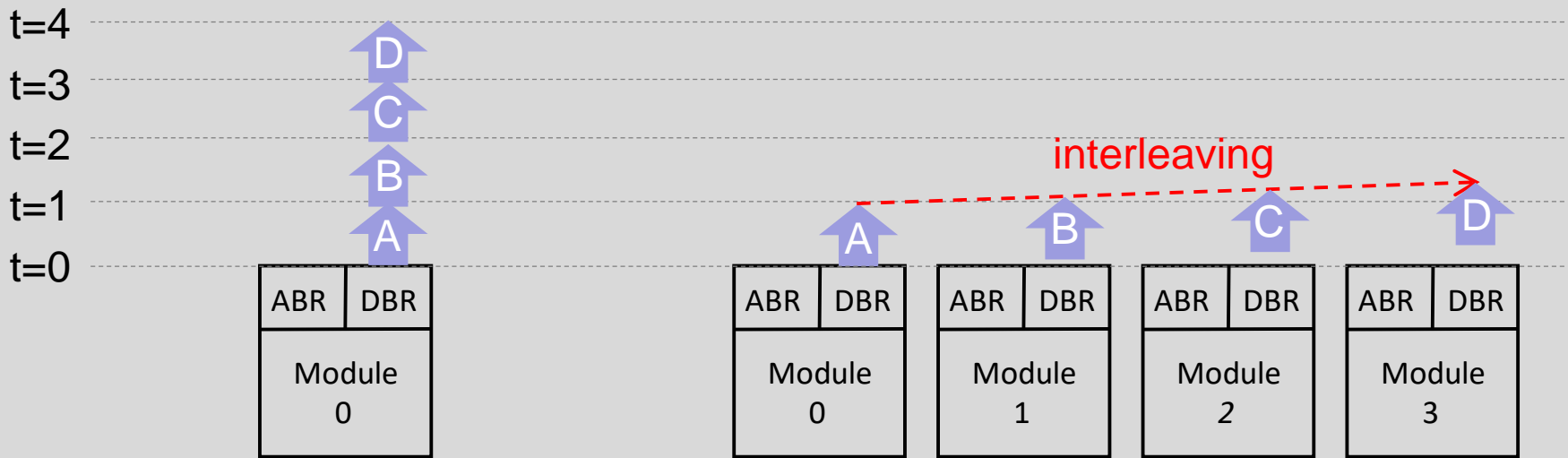

prefetch

Outline

- Performance Evaluation
  - Cache Hit/Miss Rate and Miss Penalty
  - Average Memory Access Time

- Performance Enhancements (Optional)
  - Prefetch
  - Memory Module Interleaving
  - Load-Through
  - Write Buffer

- How to reduce the miss penalty?

  – The main memory is slow in essence ...

- **Idea**: Hide the memory access latency by interleaving memory accesses across several memory modules.

  – Each module has own **Address Buffer Register (ABR)** and **Data Buffer Register (DBR)** to access memory contents.
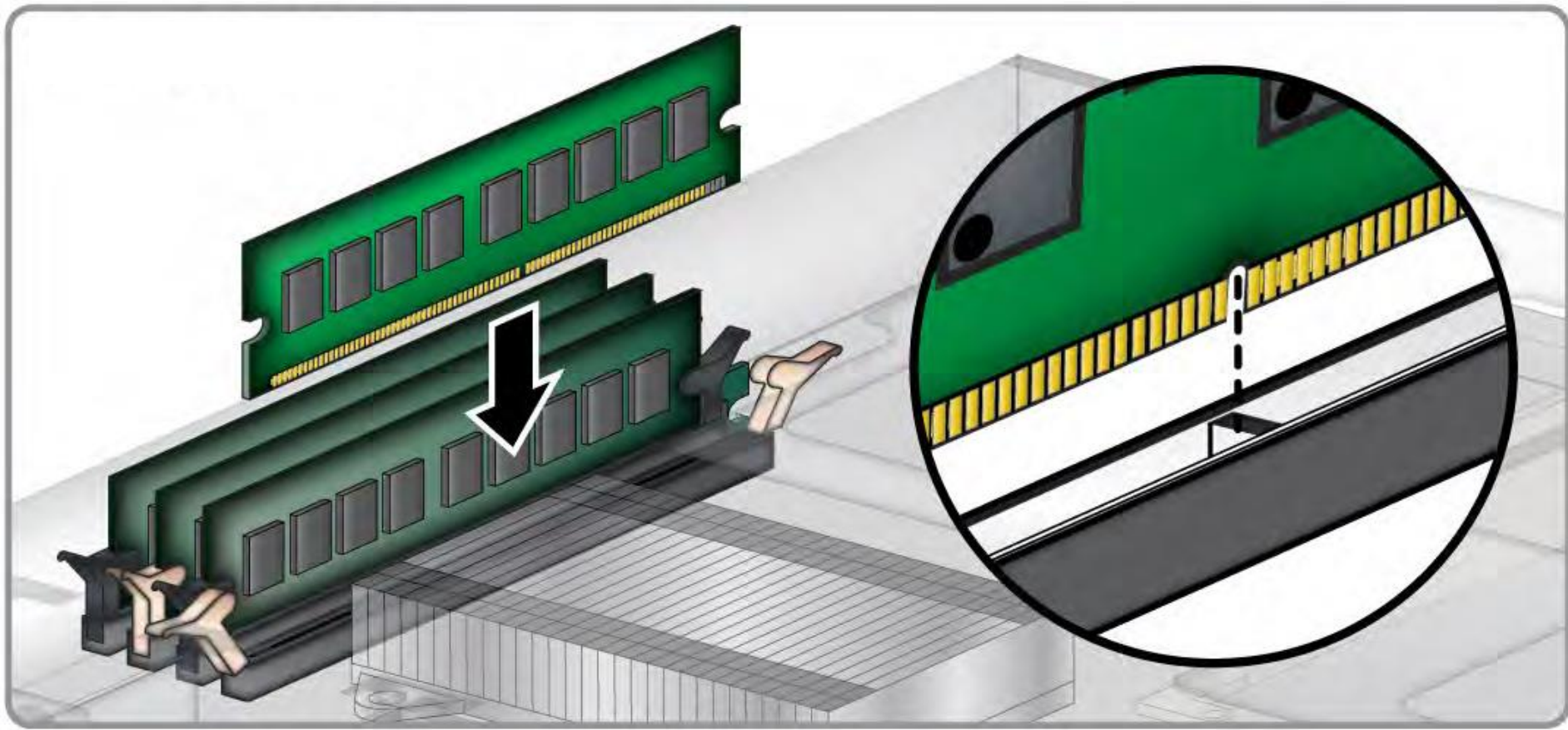

Without Memory Module Interleaving

With Memory Module Interleaving

- Multiple memory modules (usually a multiple of 2) can be installed in modern computers.
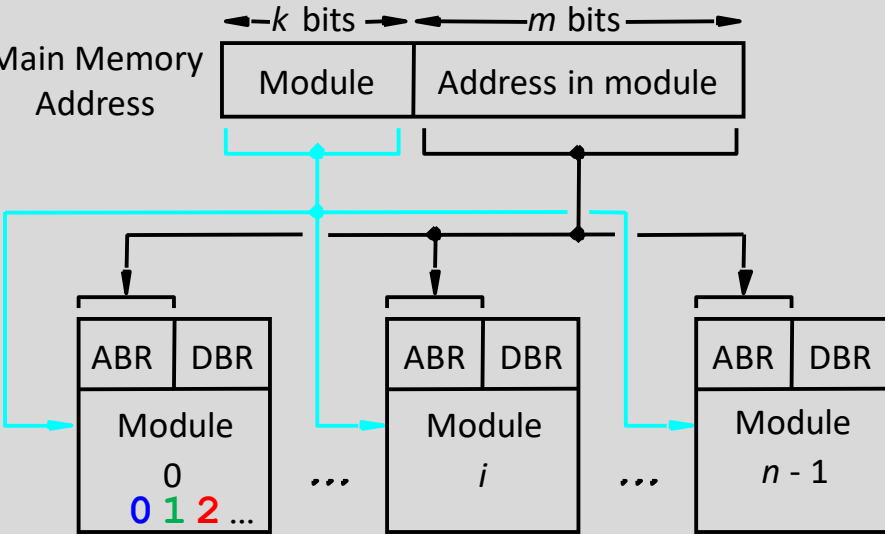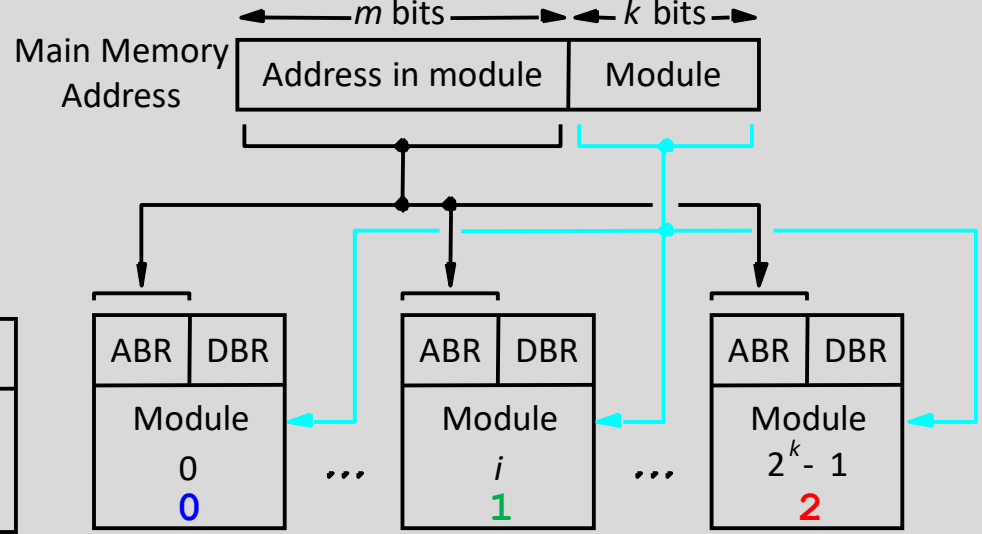
- Which scheme below can be better interleaved?
  - **Scheme (a)**: Consecutive words in the same module.
  - **Scheme (b)**: Consecutive words in successive module.
    - Keep multiple modules busy at any on time.

$(0 \ldots 0\,0\,0 \quad 0\,0\,0\,0 \ldots 0\,0\,1\,0)_2 = (2)_{10}$
$(0 \ldots 0\,0\,0 \quad 0\,0\,0\,0 \ldots 0\,0\,0\,1)_2 = (1)_{10}$
$(0 \ldots 0\,0\,0 \quad 0\,0\,0\,0 \ldots 0\,0\,0\,0)_2 = (0)_{10}$

$(0 \ldots 0\,0\,0 \quad 0\,0\,0\,0 \ldots 0\,0\,1\,0)_2 = (2)_{10}$
$(0 \ldots 0\,0\,0 \quad 0\,0\,0\,0 \ldots 0\,0\,0\,1)_2 = (1)_{10}$
$(0 \ldots 0\,0\,0 \quad 0\,0\,0\,0 \ldots 0\,0\,0\,0)_2 = (0)_{10}$

(a) Consecutive words in the **same** module

(b) Consecutive words in **successive** modules

Example of Memory Module Interleaving

- Consider a cache read miss, and we need to load a block of 8 words from main memory to the cache.

- Assume consecutive words will be assigned to successive modules for the better interleaving.

- For every memory module:
  - Address Buffer Register & Data Buffer Register
  - Module Operations:

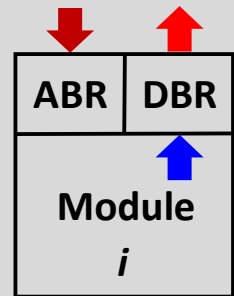   ⬇ Send an address to ABR: **1** cycle

    ⬆ Read the first word from module into DBR: **6** cycles

    ⬆ Read a subsequent word from module into DBR: **4** cycles

    ⬆ Read the data from DBR: **1** cycle

    - *Assume these operations can be performed in parallel (i.e. concurrently).*

ABR	DBR
Module *i*	

- It only allows to access (⬇ or ⬆) one module at a time.

- Total cycles to read a single word from the module:

1	6	1

- – **1** cycle to send the address
- – **6** cycles to read the first word
- – **1** cycle to read the data from DBR → $1 + 6 + 1 = 8\ cycles$

- Total cycles to read a 8-word block from the module:

Cycle 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	...	36

1st : | 1 | 6 | 1 |

$$1 + 6 + 4 \times 7 + 1$$
$$= 36\ cycles$$

(read the 1st word) 2nd : | 1 | 4 | 1 |

| 1 | 4 | 1 | 3rd

Send an address
+ (in paralle)
Read a word
from module

| ABR | DBR |
| **Module** i |

| ABR | DBR |
| **Module** i |

Read data from DBR
+ (in parallel)
Read a word
from module

8th | 1 | 4 | 1 |

With Interleaving
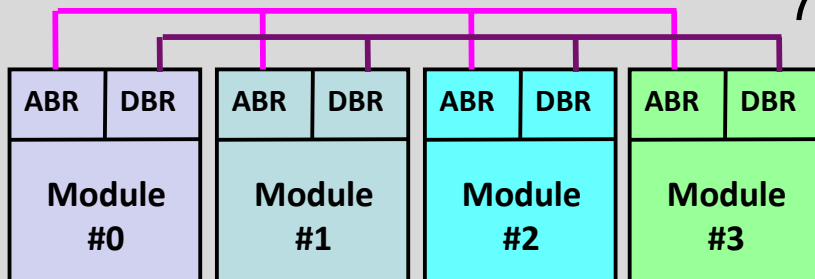
⬇ Send an address to ABR: **1** cycle
⬆ Read the first word: **6** cycles
⬆ Read a subsequent word: **4** cycles
⬆ Read the data from DBR: **1** cycle

- Total cycles to read a 8-word block from **four** interleaved memory modules:

Cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1st from #0	1				6			1							
2nd from #1	1					6		1							
3rd from #2	1						6		1						
4th from #3	1							6		1					
5th from #0				1				4			1				
6th from #1					1				4			1			
7th from #2						1				4			1		
8th from #3							1				4				1

*Send an address &
Read a word from the module #0

*Read data from DBR &
Read a word from module #0

It only allows to send the memory addresses to the modules one by one. Why? The bus is shared.

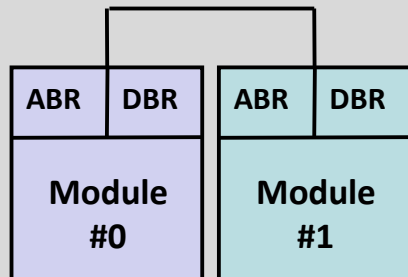| ABR	DBR	ABR	DBR	ABR	DBR	ABR	DBR
Module #0		Module #1		Module #2		Module #3	

$$1 + 6 + 1 \times 8 = 15 \; cycles$$
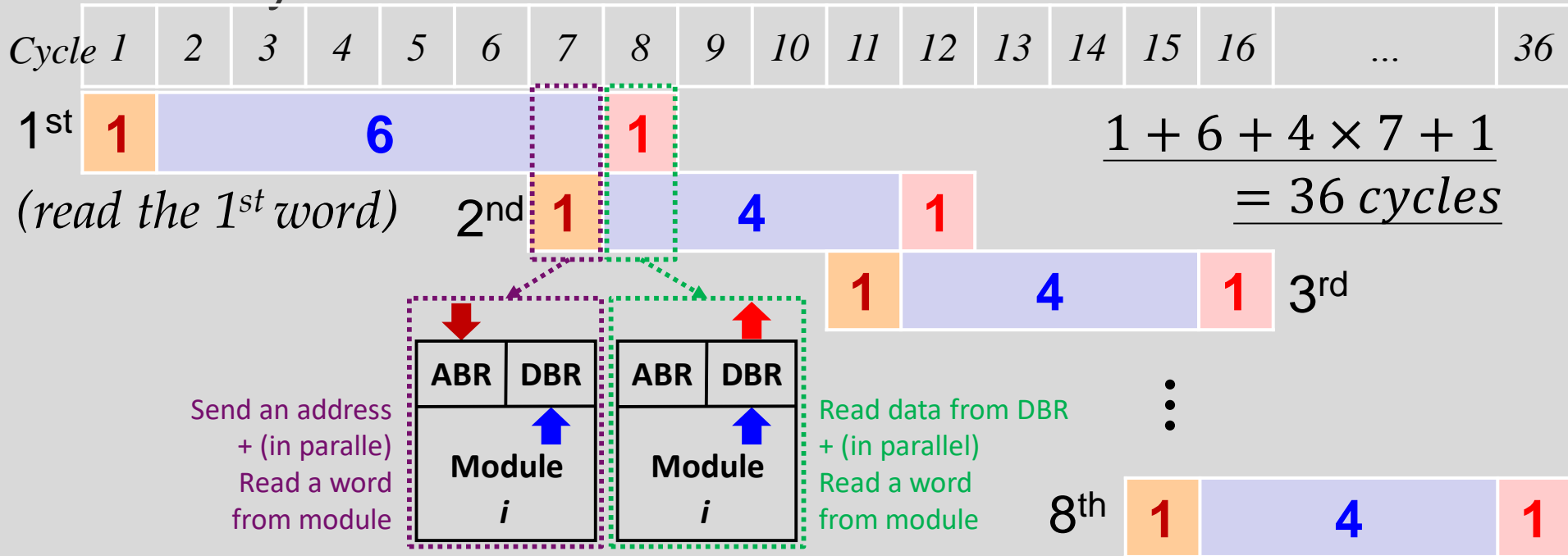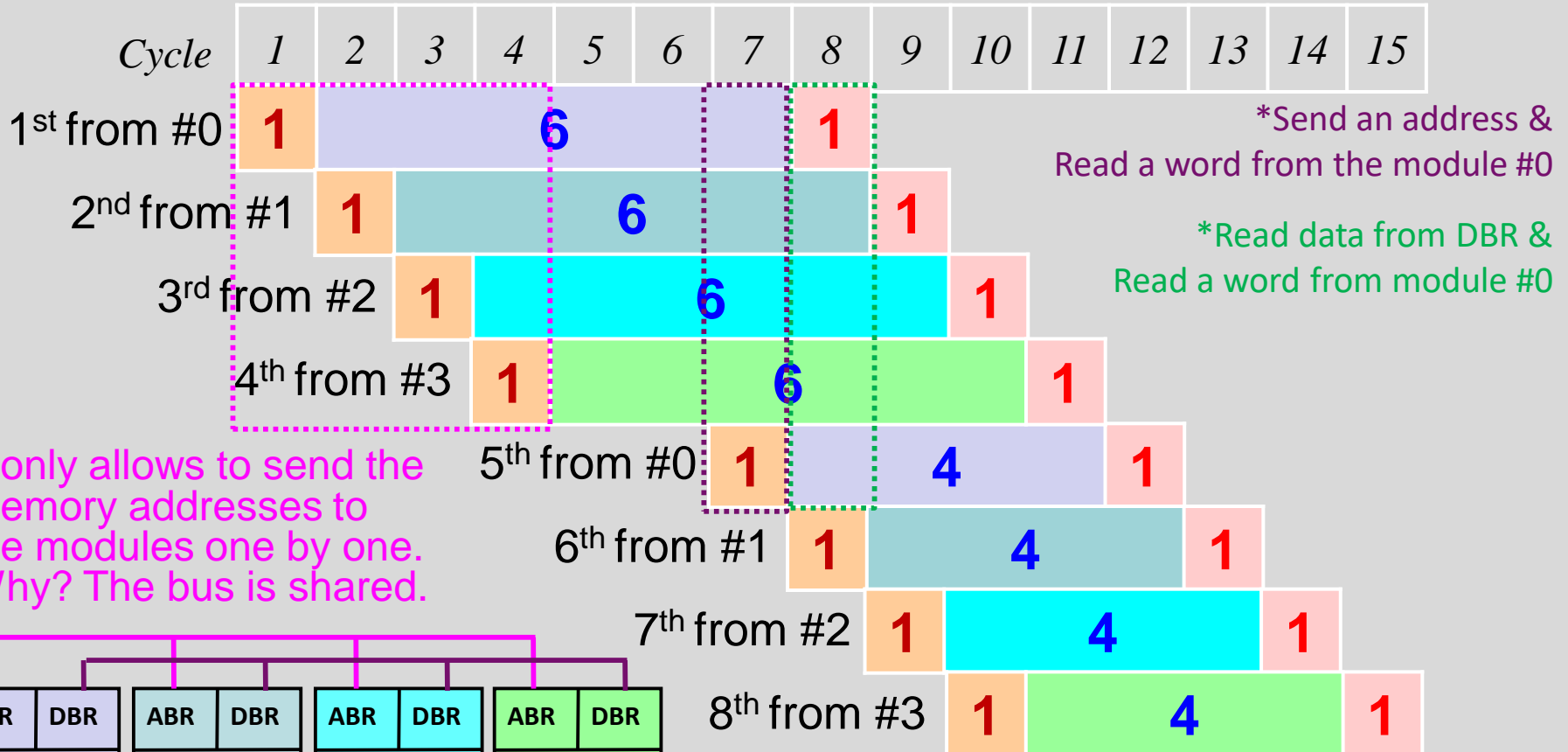
- What is the number of total cycles to read a 8-word block from **two** interleaved memory modules?

⬇ Send an address to ABR: **1** cycle
⬆ Read the first word: **6** cycles
⬆ Read a subsequent word: **4** cycles
⬆ Read the data from DBR: **1** cycle

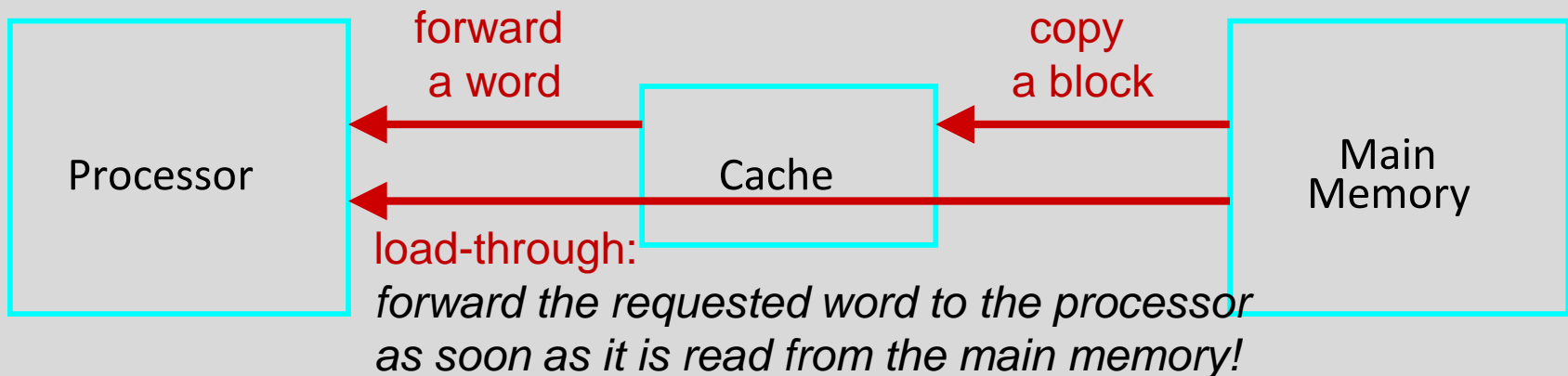| ABR	DBR	ABR	DBR
Module #0		Module #1	

Outline

- Performance Evaluation
  - Cache Hit/Miss Rate and Miss Penalty
  - Average Memory Access Time

- Performance Enhancements (Optional)
  - Prefetch
  - Memory Module Interleaving
  - Load-Through
  - Write Buffer

Load-through

- Consider a read cache miss:
  - Copy the block containing the requested word to the cache.
  - Then forward to CPU after the entire block is loaded.

- **Load-through**: Instead of waiting the whole block to be transferred, send the requested word to the processor as soon as it is ready.
  - Pros: Reduce the processor's waiting time
  - Cons: At the expense of more complex circuitry ($)

forward
a word

copy
a block

Processor

Cache

Main
Memory

load-through:
forward the requested word to the processor
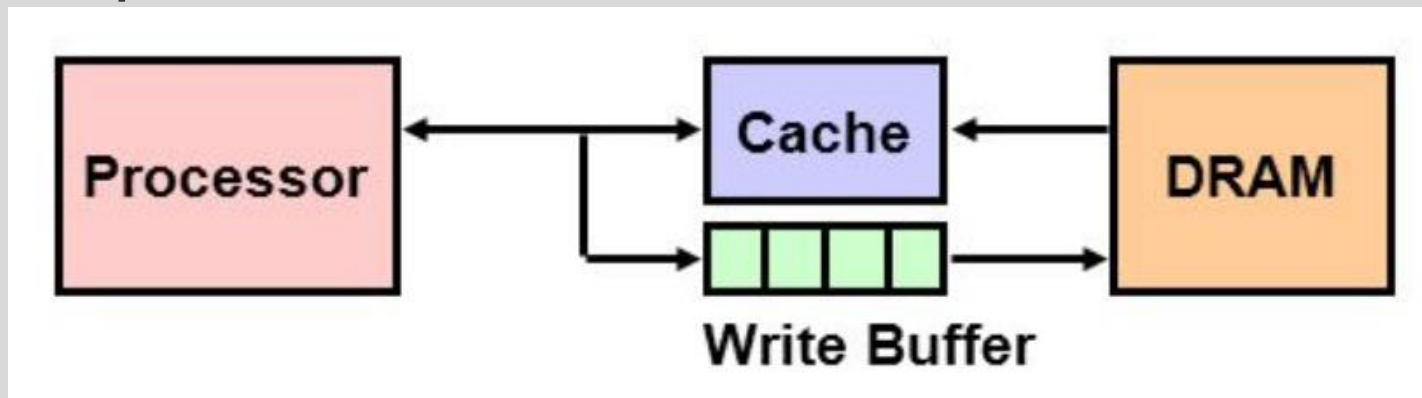*as soon as it is read from the main memory!*

- Performance Evaluation
  - Cache Hit/Miss Rate and Miss Penalty
  - Average Memory Access Time

- Performance Enhancements (Optional)
  - Prefetch
  - Memory Module Interleaving
  - Load-Through
  - Write Buffer

Write Buffer

- Observation**:** The processor typically does not need immediate access to the result of a Write operation.
- **Write Buffer**: Put writes into buffer instead of cache!
  - Read request is always served first.
  - Write request is first stored in write buffer, and then sent to memory only when there is no read request.
- Overhead: The addresses of a read request should be compared with the addresses of the write buffer.

Summary

- Performance Evaluation
  - Cache Hit/Miss Rate and Miss Penalty
  - Average Memory Access Time

- Performance Enhancements (Optional)
  - Prefetch
  - Memory Module Interleaving
  - Load-Through
  - Write Buffer