CACHE DESIGN (II)

Lecturer: Hui Annie Guo

h.guo@unsw.edu.au

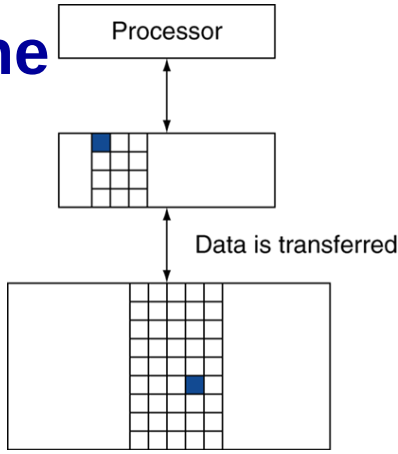
K17-501F

Lecture overview

- **Topics**
  - **Cache operations and control**
    - **Block placement**
    - **Block identification**
    - **Block replacement**
    - **Write strategy**
- **Suggested reading**
  - **H&P Chapter 5.4**

Recall: Cache (cont.)

- For control of the operation of cache, four issues should be addressed
  - Where to put a memory block in cache?
    - Block placement strategy
  - How to find a memory block in cache?
    - Block identification
  - If there are no free spaces in cache, which block can be replaced by a new memory block?
    - Block replacement
  - When memory data is updated, how is cache involved in write?
    - Write strategy

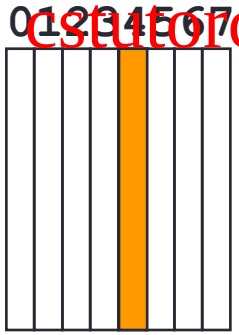


Block placement

- How to determine the cache location for a memory block?
- Based on memory block address and cache structures:
  - Direct mapped
    - A memory block can be placed in one and only one location in the cache
  - Fully associative
    - A memory block can be placed in any location in the cache
  - Set associative
    - A memory block can be placed in any location in a set (one and only one) of the cache

Example (1)

- **For a memory block with the block address 28, where can it be placed in a direct mapped cache of 8 blocks?**

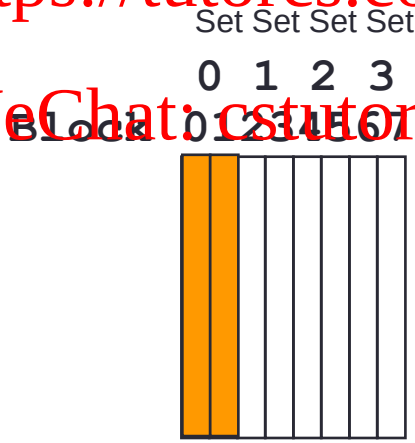


Example (2)

- **For a memory block with the block address 28, where can it placed in a fully associative cache of 8 blocks?**

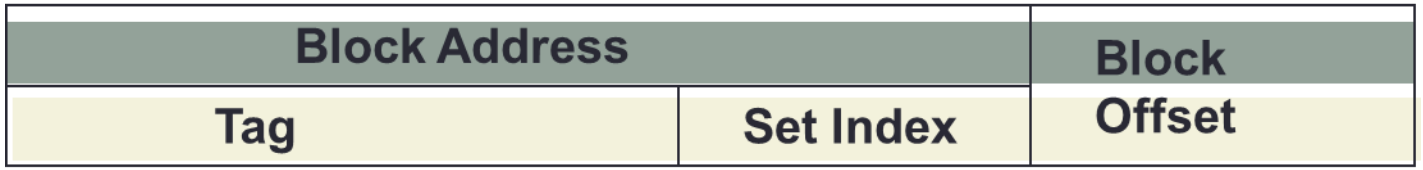
Example (3)

- **For a memory block with the block address 28, where can it placed in a 2-way set associative cache of 8 blocks?**



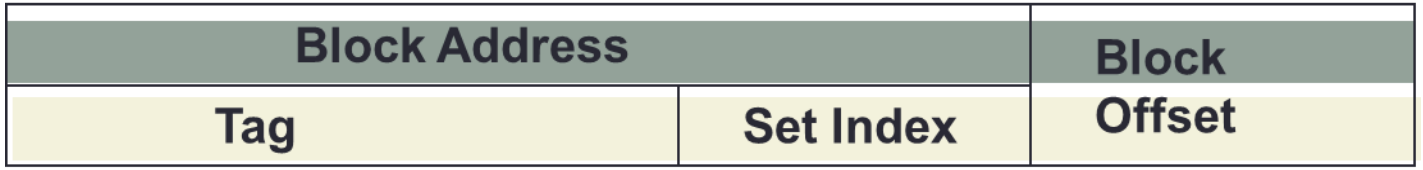
Block identification

- **How to identify whether a memory block is cached? And where?**
  - **Based on the memory block address**
    - **Tag field**
    - **Set-index field**
      - **Specifying the possible cache locations**
      - **Field size**
        - **0 bit → fully associative cache**
        - **n bit → 2^n sets**
  - **If the entry with the matched tag is found and is valid, the block is cached.**



In-class exercise (1)

- **Given the memory address format as shown below and a cache of *B* blocks,**
  - **(a) What is the minimal size of Set Index?**
  - **(b) What is the maximum size of Set Index?**



In-class exercise (2)

- **A 4-way set associative cache contains 2^n blocks.**
  - **How many bits are used to index a memory block in the cache?**

Block replacement

- When cache is full, which cache block is replaced by the new memory block?
  - For direct mapped
    - Only one option
      - Fixed location
      - The related cache block is always replaced
  - For set associative or fully associative
    - Multiple options
    - Two typical approaches
      - Random
        - Randomly select one of the multiple optional locations for replacement
      - LRU (Least Recently Used)
        - Select the block that is least recently used

In-class exercise

- **For a 4-way set associative cache,**
  - **(a) what is the probability that a block is replaced by a new memory block if the random replacement policy is used?**
  - **(b) can you determine the probability when the LRU policy is applied?**

Comparison of two replacement policies

Experiment results: cache miss rate of different cache configurations

associ. / policy / cache size	2-way		4-way		8-way	
	LRU	Random	LRU	Random	LRU	Random
16 KB	5.2%	5.7%	4.7%	5.3%	4.4%	5.0%
64 KB	1.9%	2.0%	1.5%	1.7%	1.4%	1.5%
256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

- What conclusion can you draw from the experiment data?

Write strategy

- When memory data is updated, how is the cache affected? How is the data saved?
- Four write policies
  - When the write has a cache hit
    - Write though
    - Write back
  - When the write has a cache miss
    - Write allocate
    - Write not allocate

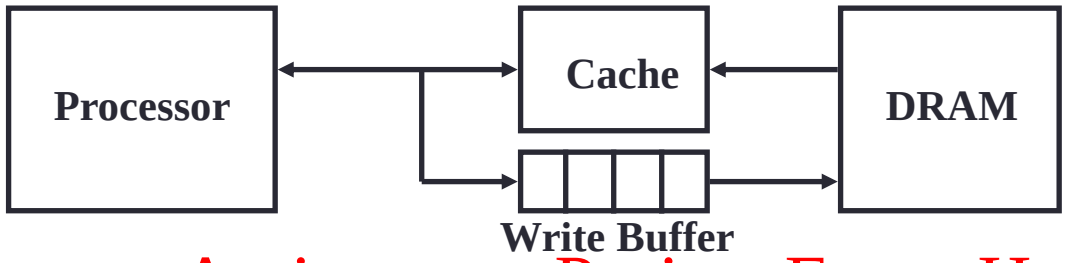
Policies on write hit

- **Write through (WT)**
  - **Data is written to both cache and memory.**
- **Write back (WB)**
  - **Data is written only to cache.**
  - **The modified cache block is written to memory only when it is replaced.**
  - **Additional dirty bit is required.**
    - **A block is dirty if data in the block has been modified; otherwise, the block is clean and no need to be written back when it is replaced.**

Policies on write hit (cont.)

- **Pros and Cons of WT and WB**
  - **WT**
    - **+ read misses don't result in writes**
    - **- potentially high memory traffic**
      - **performance degradation**
  - **WB**
    - **+ less memory accesses**
      - **no memory writes needed for repeated processor writes**
    - **- cache coherence issue**
      - **to be discussed in the multiprocessor design**

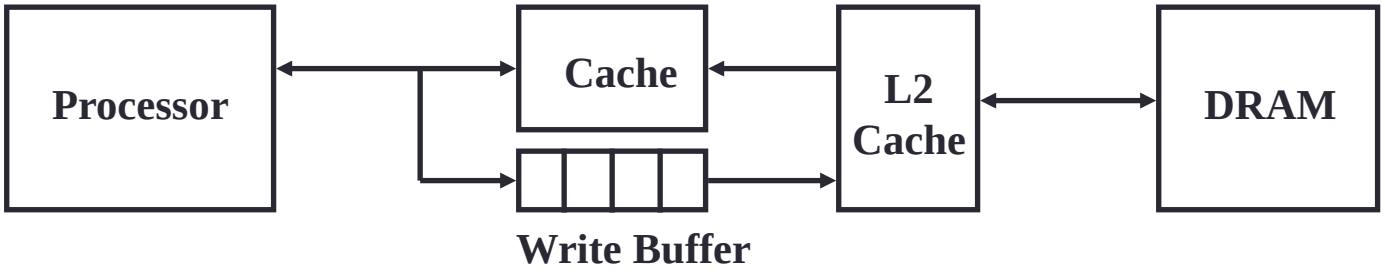
Improve design with write buffer



- To reduce the impact of slow memory access on the performance, a write buffer can be used
- For a write-to-memory operation,
  - Processor: writes data to the write buffer
  - Memory controller: writes contents of the buffer to memory
- Write buffer is just a FIFO (First In First Out)
  - Small, typical number of entries: 4
  - Works fine if processor write frequency << 1 / DRAM write cycle.

Improve design with write buffer (cont.)

- **If processor write frequency > 1 / DRAM write cycle,**
  - **Write buffer will overflow (aka write buffer saturation)**
- **Solution for write buffer saturation**
  - **Add a second level (L2) cache**



Policies on write miss

- **Write Allocate**
  - **Allocates a cache block in cache for the write operation and has two ways**
    - **Fetch-on-write**
      - **Read in the whole block from the memory to the cache, then write the new data to the cache block**
    - **Not-fetch-on-write**
      - **Write to cache immediately**
      - **The cache block is invalid, except for the data word that is written**
- **Write Not Allocate**
  - **No block in the cache is used for the data written**
    - **write misses direct go to the next lower level.**

Types of cache misses

- Cache performance is closely related to the cache misses.
- There are three types cache misses
  - Compulsory miss (aka cold start miss)
    - First access to a block and the block has never been cached.
  - Conflict miss (aka collision miss)
    - In a non-fully associative cache
    - Due to competition for an entry in a set
  - Capacity miss
    - Due to limited cache size
    - Related to fully associative cache

Recall: Example

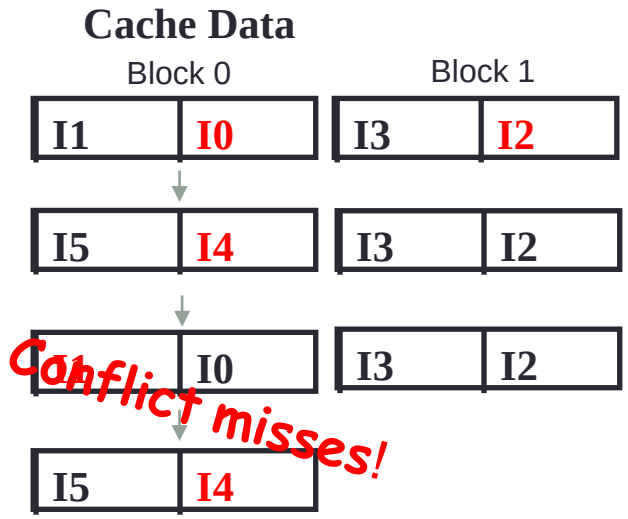
- **The following memory locations need to be accessed in sequence: 10110, 11010, 10000, 00010. How is the cache updated?**
  - **After power on**

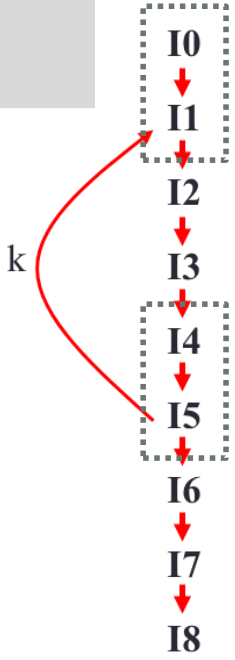
Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

All are compulsory misses!

Recall: Class exercise

- **If the cache shown in the previous slide is re-structured into two blocks with the same cache size, how are the cache contents changed?**





Trade-offs in the cache design

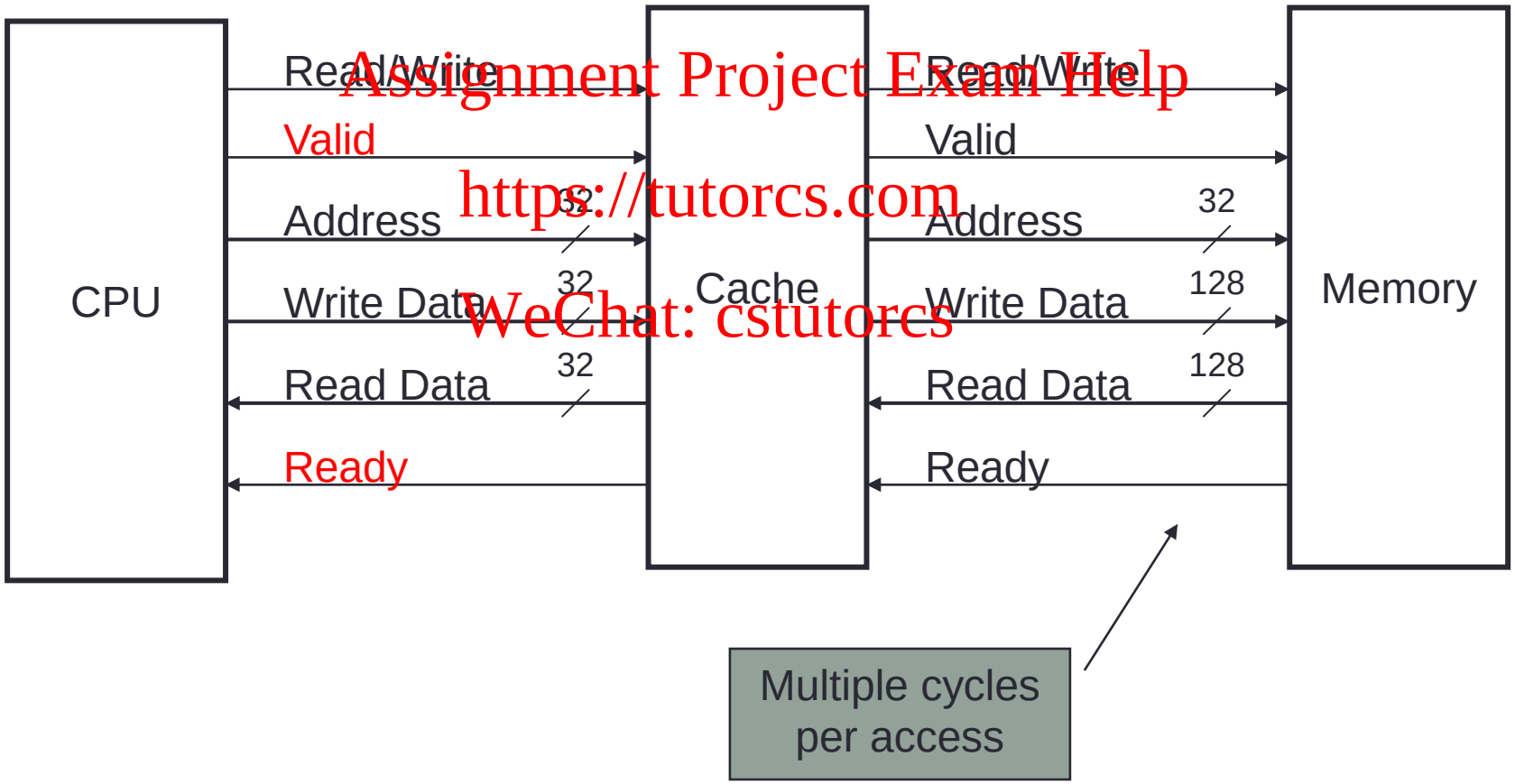
- **Impact of cache size, associativity, and block size on cache performance**

Design change	Effect on miss rate	Negative performance effect
Increase cache size	Decrease capacity misses	May increase hit time
Increase associativity	Decrease conflict misses	May increase hit time
Increase block size	Decrease compulsory misses	Increases miss penalty May increase conflict misses

- **Trade-off should be played in the cache design**

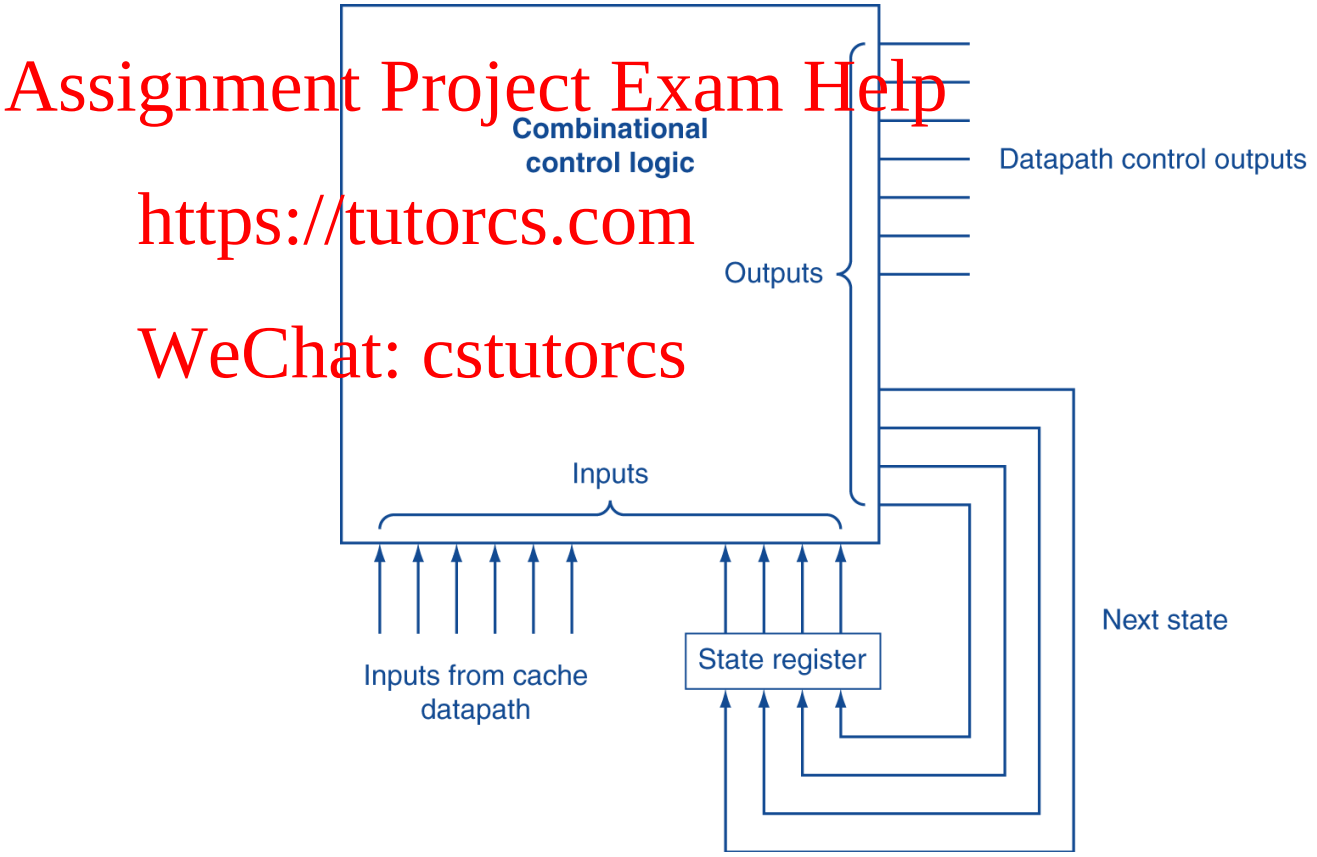
Cache interface

- **A typical example**

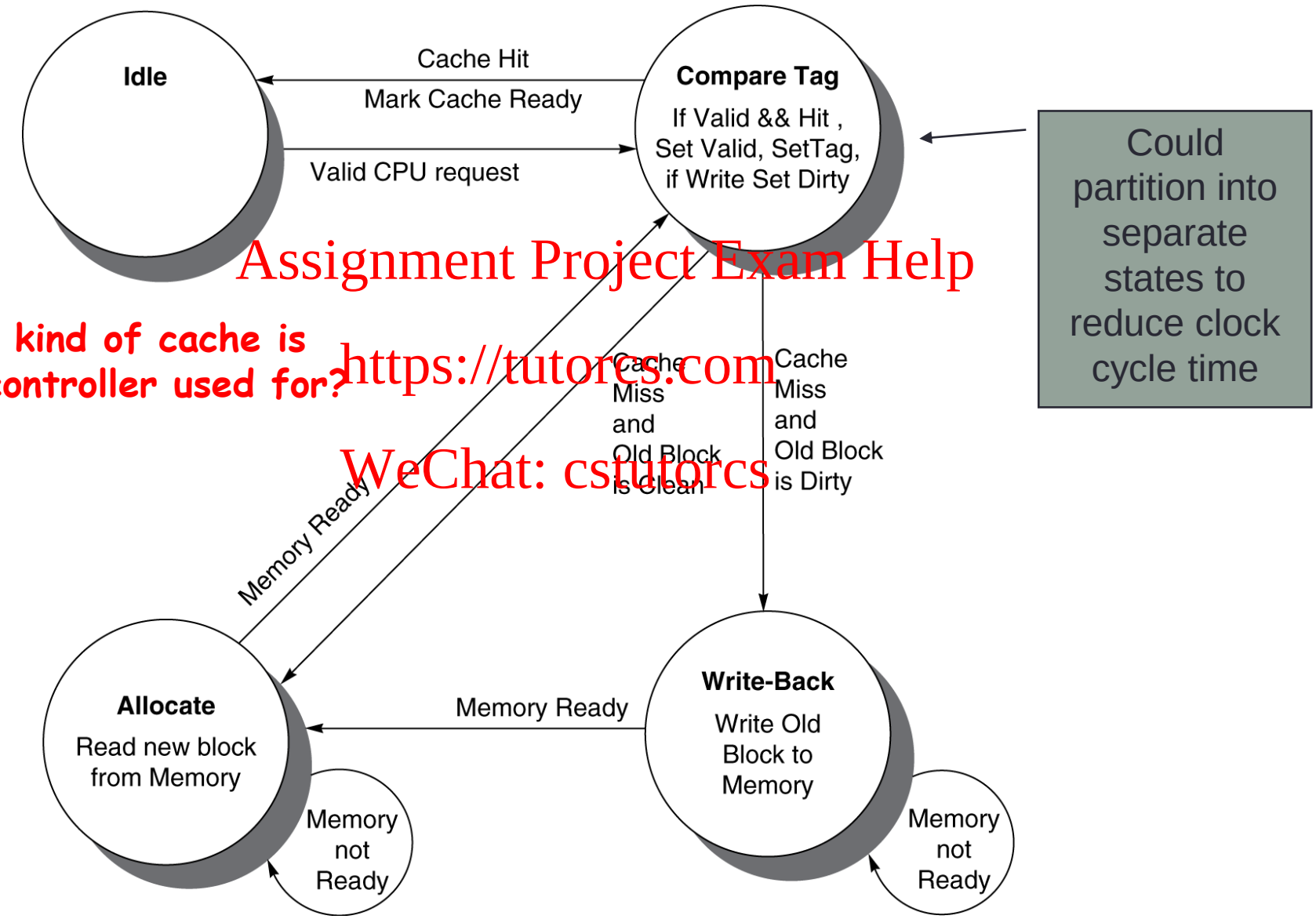


Cache controller

- **Handles the memory read/write request**
  - **By using a FSM**



Cache controller FSM – an example



What kind of cache is this controller used for?

Recall: Impact of the speed gap on performance

- Suppose a processor executes at
  - clock rate = 2 GHz
  - CPI = 1.1
  - 50% arith/logic, 30% ld/st, 20% control
- Suppose data memory operations get 50 cycle penalty
  - Pipeline has to wait 50 cycles for each memory access
- CPI
  - = ideal CPI + average stalls per instruction

  = 1.1 + 0.30 x 50

  = 16.1
- Because of the slowness of memory, on average, the pipeline outputs every 16 clock cycles!

Improvement on Performance with cache

- **Suppose a processor executes at**
  - **Clock Rate = 200 MHz (5 ns per cycle)**
  - **CPI = 1.1**
  - **50% arith/logic, 30% ld/st, 20% control**
- **Suppose that 10% of data memory operations get 50 cycle miss penalty**
  - **Pipeline has to wait 50 cycles for each memory access**
- **CPI**
  - **= ideal CPI + average stalls per instruction**

    = 1.1 + 0.1 x 0.30 x 50

    = 1.1 cycle + 1.5 cycle

    = 2. 6
- **The performance is improved.**

In-class exercise

Given a sequence of word address references (written in decimal):
1, 4, 8, 5, 20, 17, 19, 9, 11, 4, 5, 6

(a) Assuming a direct-mapped cache with 8 one-word blocks and the cache is initially empty, label each reference in the list as a hit or miss and show the final contents of the cache.

Cache block	Access #1	Access #2	Access #3	Final Contents
0	8(miss)			(8)
1	1(miss)	17(miss)	9(miss)	(9)
2				N/A
3	19(miss)	11(miss)		(11)
4	4(miss)	20(miss)	4(miss)	(4)
5	5(miss)	5(hit)		(5)
6	6(miss)			(6)
7				(N/A)

In-class exercise

Given a sequence of word address references (written in decimal):
1, 4, 8, 5, 20, 17, 19, 9, 11, 4, 5, 6

(b) Show the hits and misses and the final contents for a direct-mapped with 2-word blocks and a total size of 8 words. Assume blocks are aligned with even addresses

Cache block	word	Access #1	Access #2	Access #3	Access #4	Access #5	Final contents
0	0	0(line)	8(miss)	16(line)	8(line)		8
0	1	1(miss)	9(line)	17(miss)	9(miss)		9
1	0	18(line)	10(line)				10
1	1	19(miss)	11(miss)				11
2	0	4(miss)		20(miss)	4(miss)		4
2	1	5(line)	5(hit)	21(line)	5(line)	5(hit)	5
3	0	6(miss)					6
3	1	7(line)					7

In-class exercise

Given a sequence of address references (written in decimal format): 1, 4, 8, 5, 20, 17, 19, 9, 11, 4, 5, 6

(c) Show the hits and misses and the final contents for a two-way set-associative cache with one-word blocks and a total size of 8 words. Assume LRU Replacement.

set	block	Access #1	Access #2	Access #3	Final contents
0	0	4(miss)	20(miss)		20
1	0	8(miss)	4(miss)		4
0	1	1(miss)	17(miss)	5(miss)	5
1	1	5(miss)	9(miss)		9
0	2	6(miss)			6
1	2				N/A
0	3	19(miss)			19
1	3	11(miss)			11

In-class exercise

Given a sequence of address references (written in decimal format): 1, 4, 8, 5, 20, 17, 19, 9, 11, 4, 5, 6

(d) Show the hits and misses for a fully associative cache with one-word blocks and a total size of 8 words. Assume LRU Replacement.

cache block	Access #1	Access #2	Access #3	Final contents
0	1(miss)	11(miss)		
1	4(miss)	4(hit)		
2	8(miss)	6(miss)		
3	5(miss)	5(hit)		
4	20(miss)			
5	17(miss)			
6	19(miss)			
7	9(miss)			

In-class exercise

Given a sequence of address references (written in decimal format): 1, 4, 8, 5, 20, 17, 19, 9, 11, 4, 5, 6

(e) Show the hits and misses for a fully associative cache with two-word blocks and a total size of 8 words. Assume LRU Replacement.

cache block	Access #1	Access #2	Access #3	Access #4	Final contents
0	0(line)	16(line)	4(miss)		
0	1(miss)	17(miss)	5(line)	5(hit)	
1	4(miss)		8(line)		
1	5(line)	5(hit)	9(miss)		
2	8(miss)	18(line)	6(miss)		
2	9(line)	19(miss)	7(line)		
3	20(miss)	10(line)			
3	21(line)	11(miss)			