# Lab 6 - Cache

CSED 311 Computer Architecture Lab

Sungjun Cho

2021 - 05 - 11

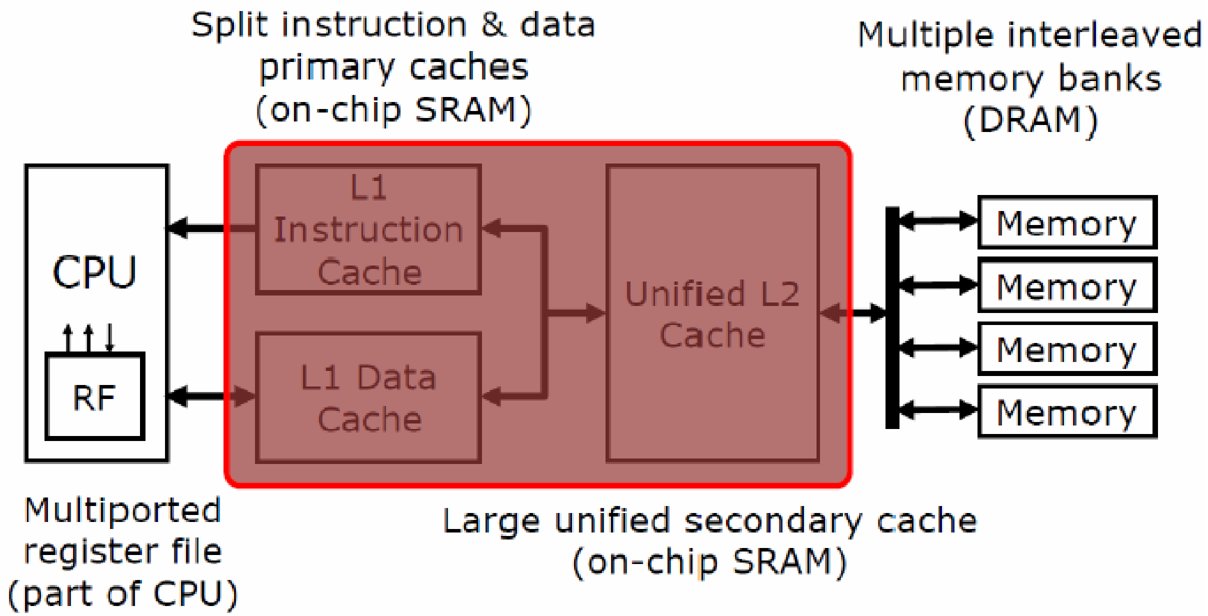
allencho1222@postech.ac.kr
csed311-ta@postech.ac.kr

# Objectives

- Understand how cache works
- Implement a set-associative cache on your pipelined CPU
- Evaluate the speed-up achieved by using cache
  - Hit ratio
  - Corresponding speed-up (vs. no-cache CPU)

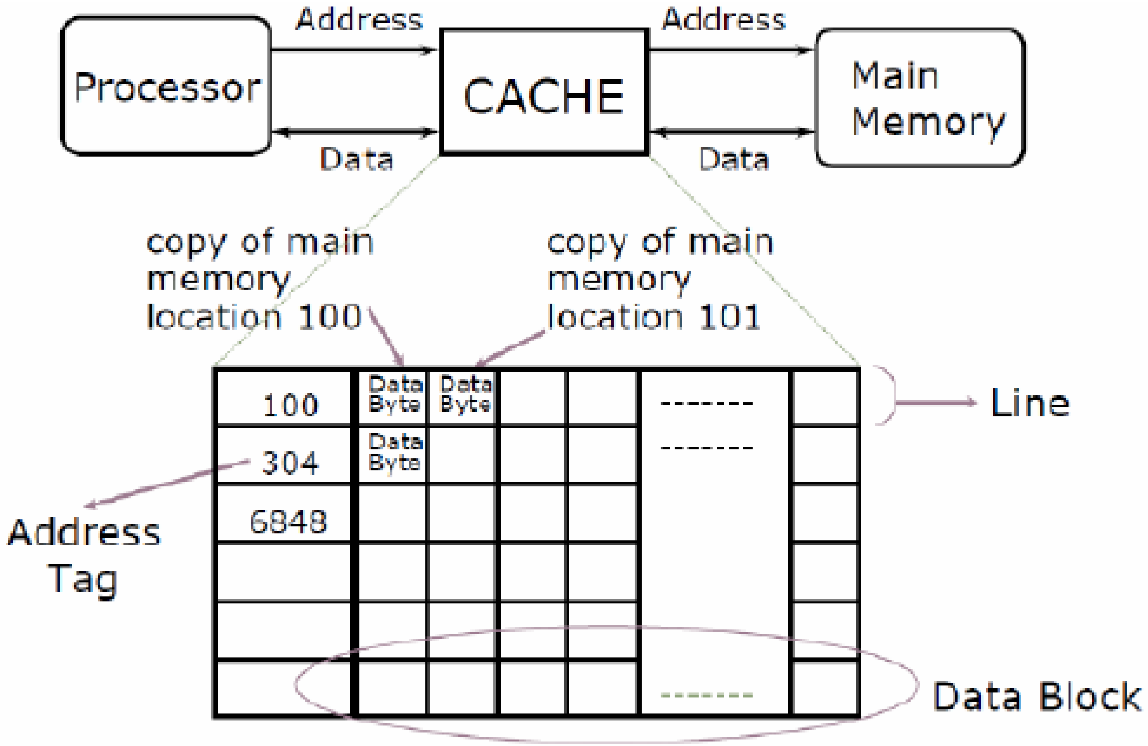
# Cache

- Mitigating the gap between CPU and memory
  - Memory access: few hundred cycles
  - Cache access: few cycles
- Why does it work?
  - Locality!



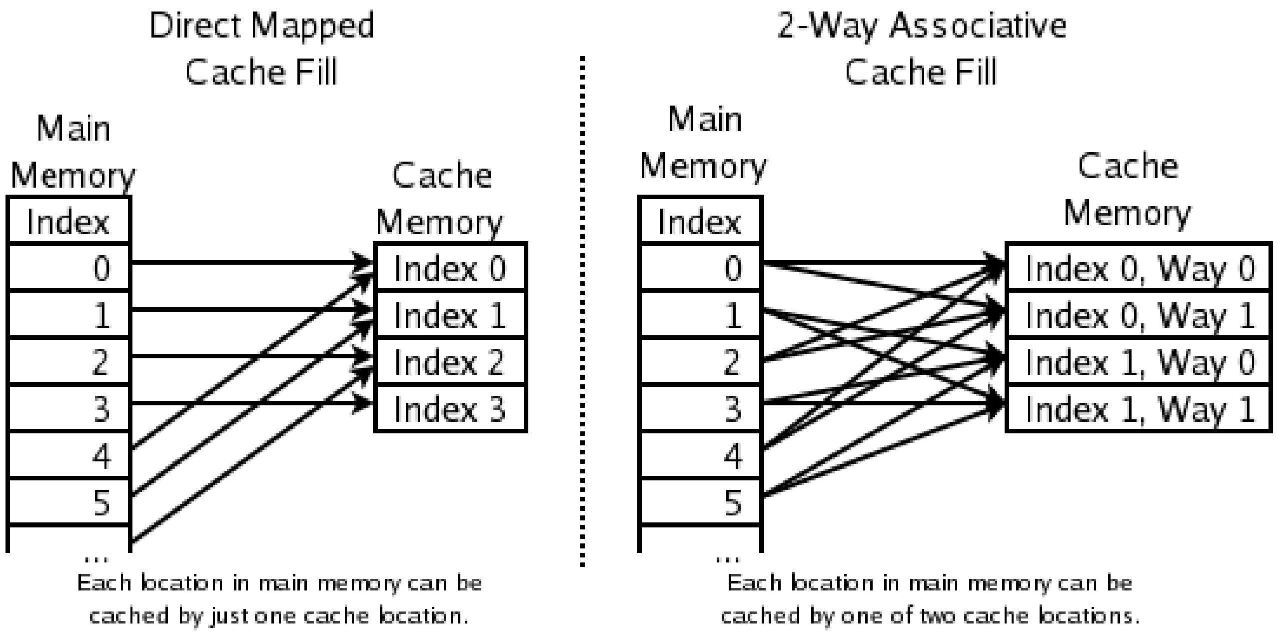
# Cache: internal structure

- Tag
  - Detect address conflicts
- Data
  - Fetch by line: exploiting spatial locality



# Cache: associativity

- Associativity
  - Reducing address conflicts
- Direct mapped, n-way, fully associative
  - Tradeoffs exist



# Cache: other design choices

- Replacement policy
  - Random, LRU, FIFO, …,
  - Each one has strengths & drawbacks
- Write policy
  - Write-through, writeback, write-no-allocate
  - Related to coherency management

# Lab Assignment 06 (1/5)

- Design and implement your own cache with the following requirements:
  - 2-way set associative, single-level cache
  - Capacity: 32 words / Line size: 4 words
  - If hit, return data in the following cycle
  - It should be a part of CPU (not TB)

- You have to choose the following design choices:
  - Replacement policy, write policy
  - Unified (32 words == 8 Lines) or separate I/D (4 lines ea.) cache

# Lab Assignment 06 (2/5)

- You may implement direct mapped cache with reduced score of 80%
  - Direct-mapped, single-level cache
  - Capacity: 16 words / Line size: 4 words
  - If hit, return data in the following cycle
  - It should be a part of CPU (not TB)

# Lab Assignment 06 (3/5)

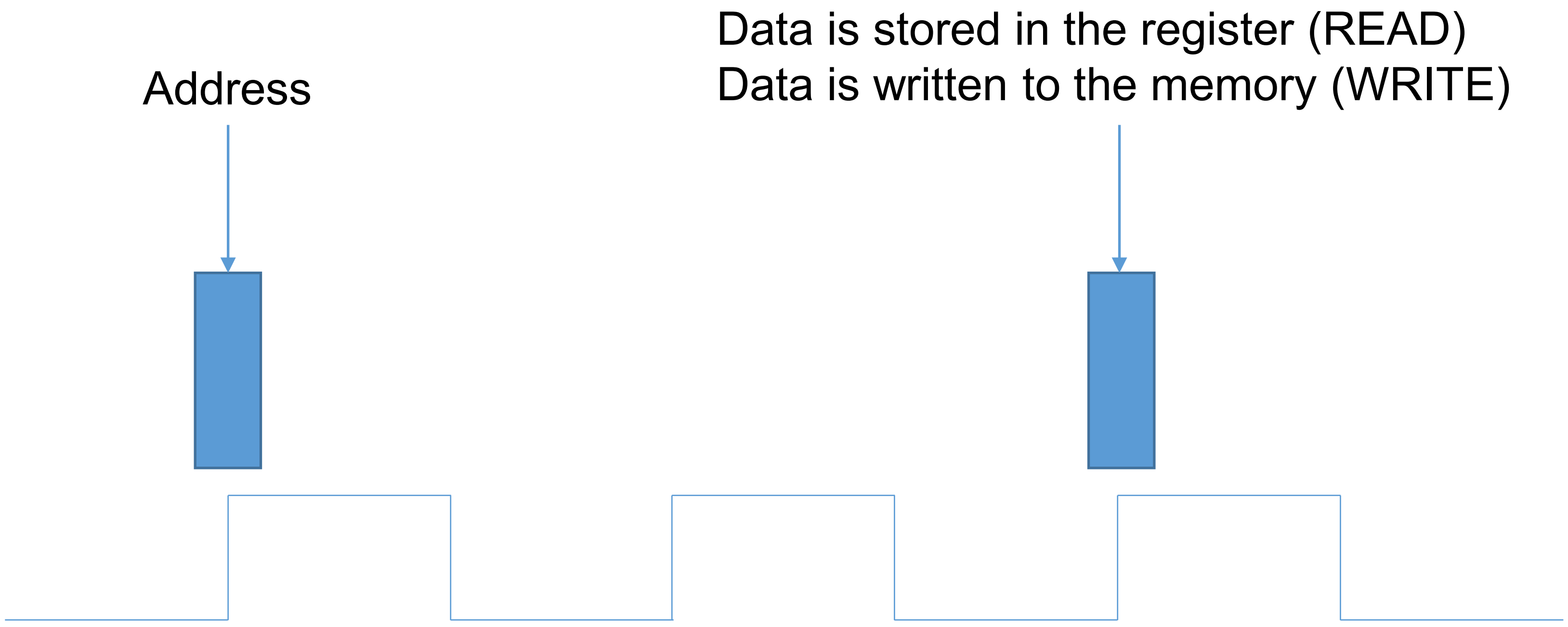
- Memory access latency – Previous pipelined CPU
  - One memory access fetches one word with one cycle

- New memory access latency
  - You're required to modify memory.v
  - Cache hit takes one cycle
  - One memory access should fetch four words into cache (= one cache line) and take six cycles
  - For your baseline CPU (CPU without cache), one memory access should fetch one word and take two cycles

  - Then, you need to implement two different memory models
    - ✓ Memory for CPU with cache: return four words in six cycles
    - ✓ Memory for CPU without cache: return one word in two cycles

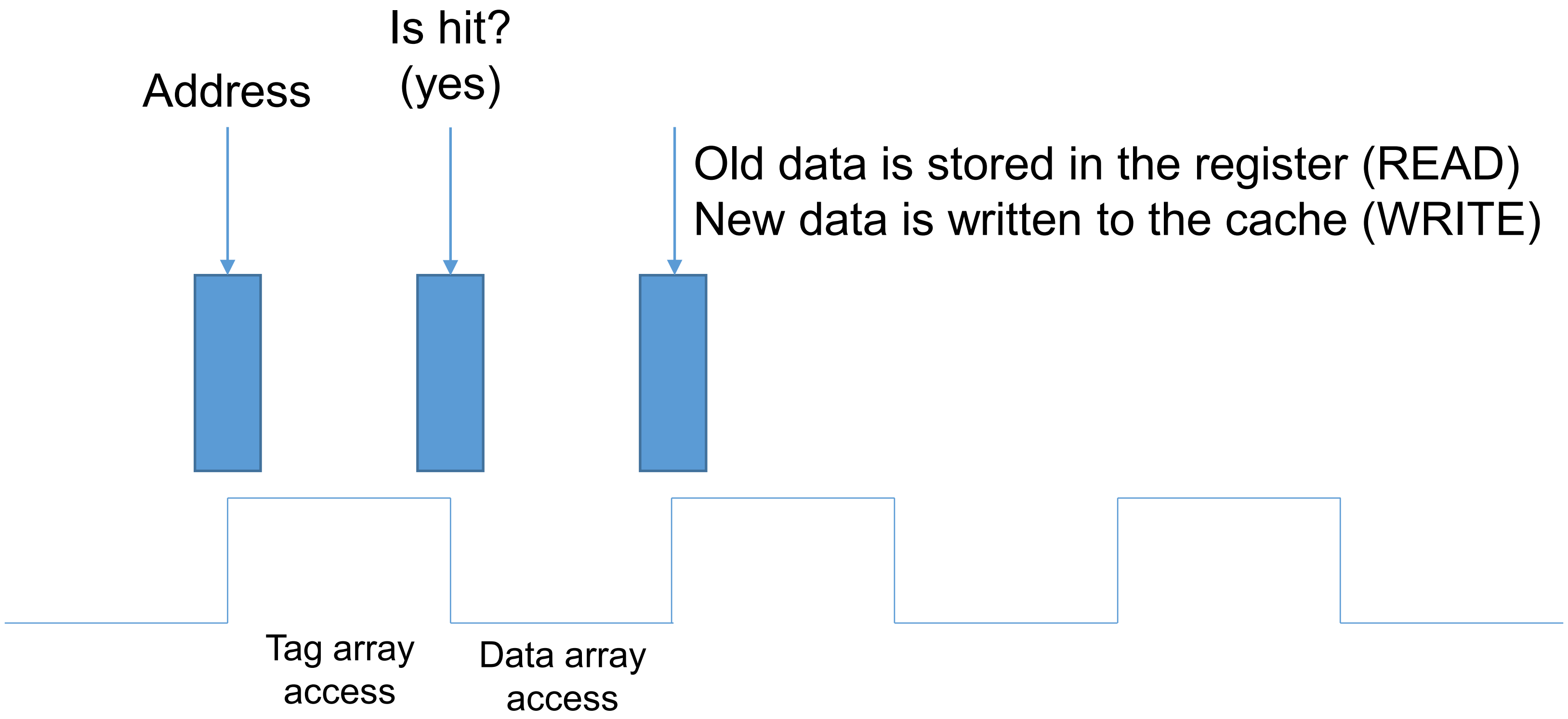
# Waveform (Baseline CPU)

- Cache does not exist

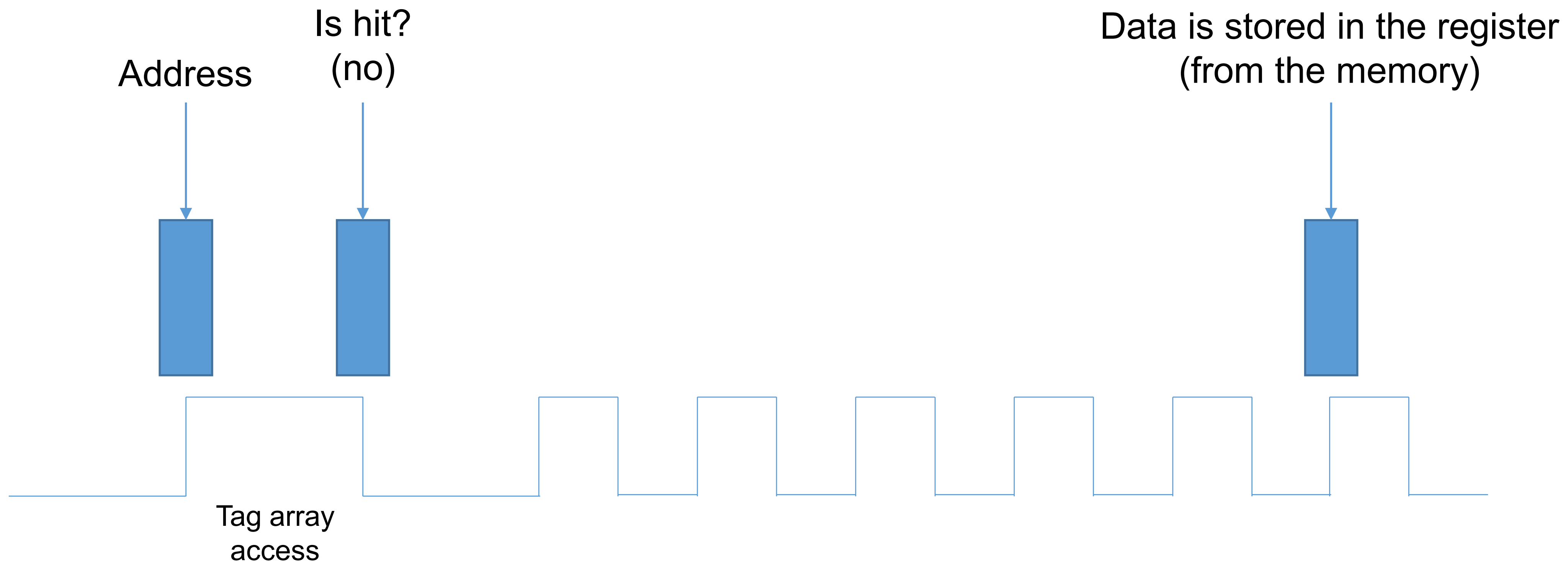


# Waveform (Cache hit)

- Cache miss/hit and data can be checked in a single cycle

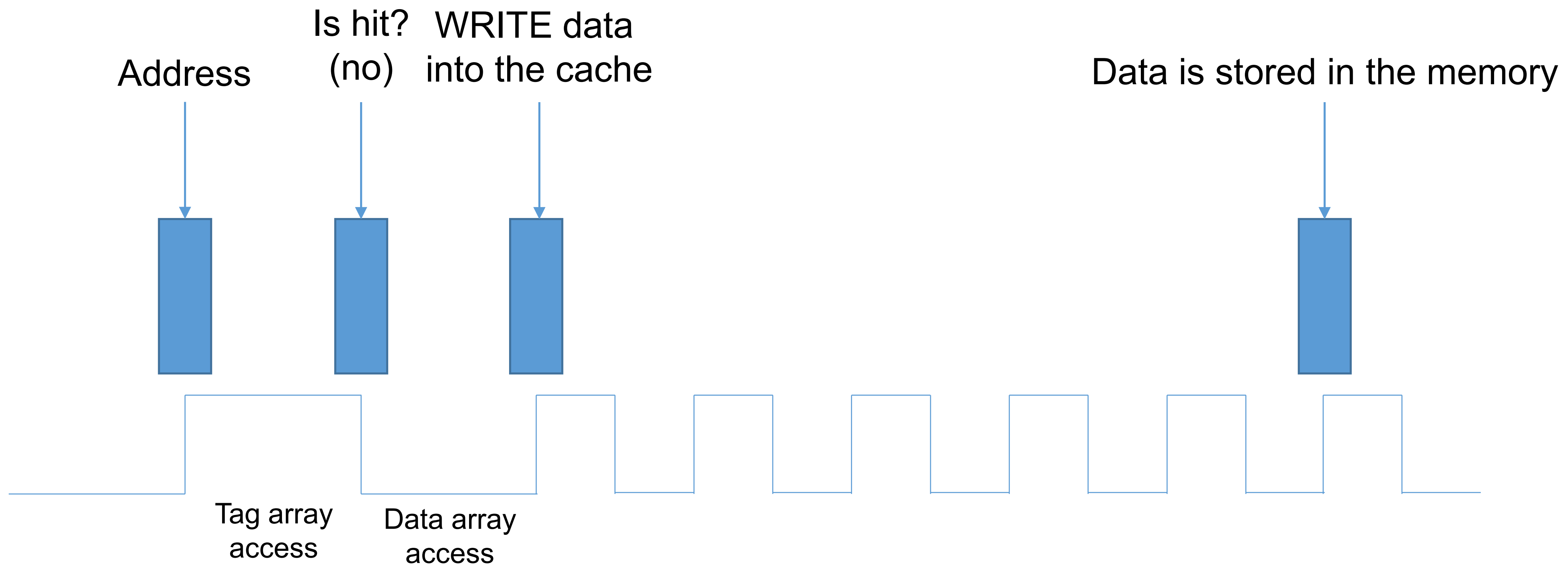


# Waveform (Cache miss)



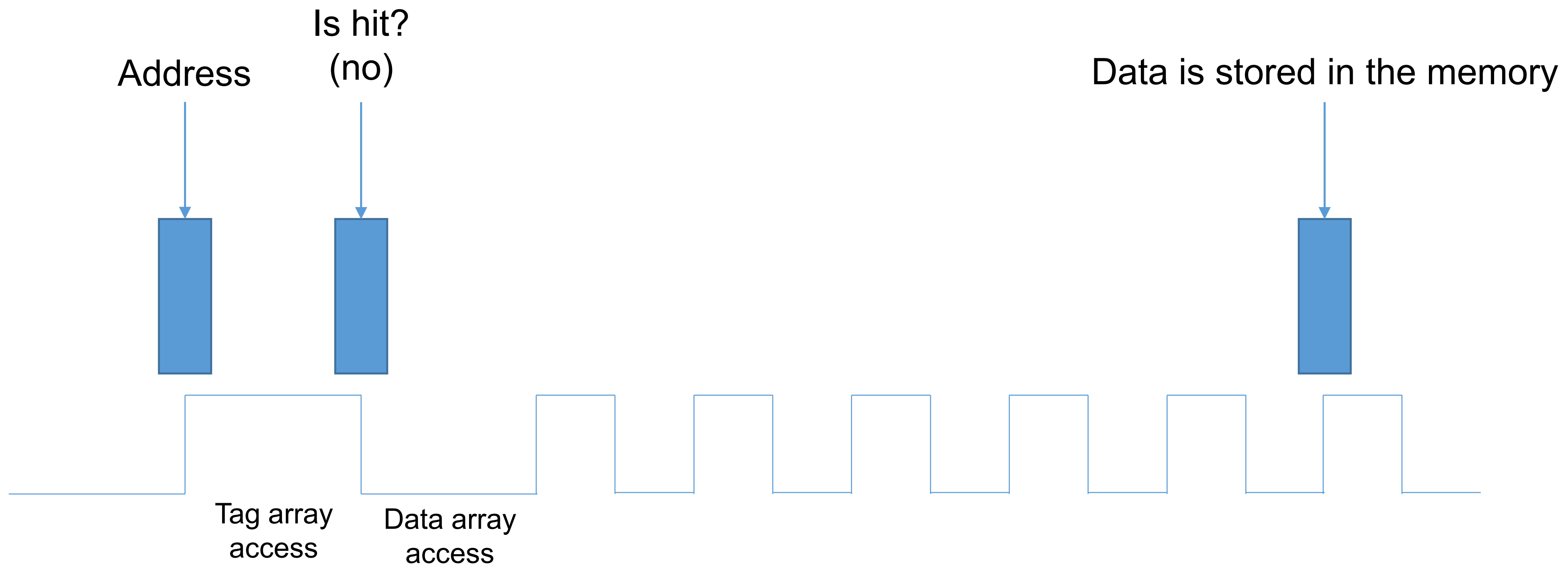
# Waveform (WRITE-THROUGH)

- Usually comes with write-no-allocate
  - Cache hit (WRITE request)
    - ✓ WRITE data into the cache + WRITE data into the memory



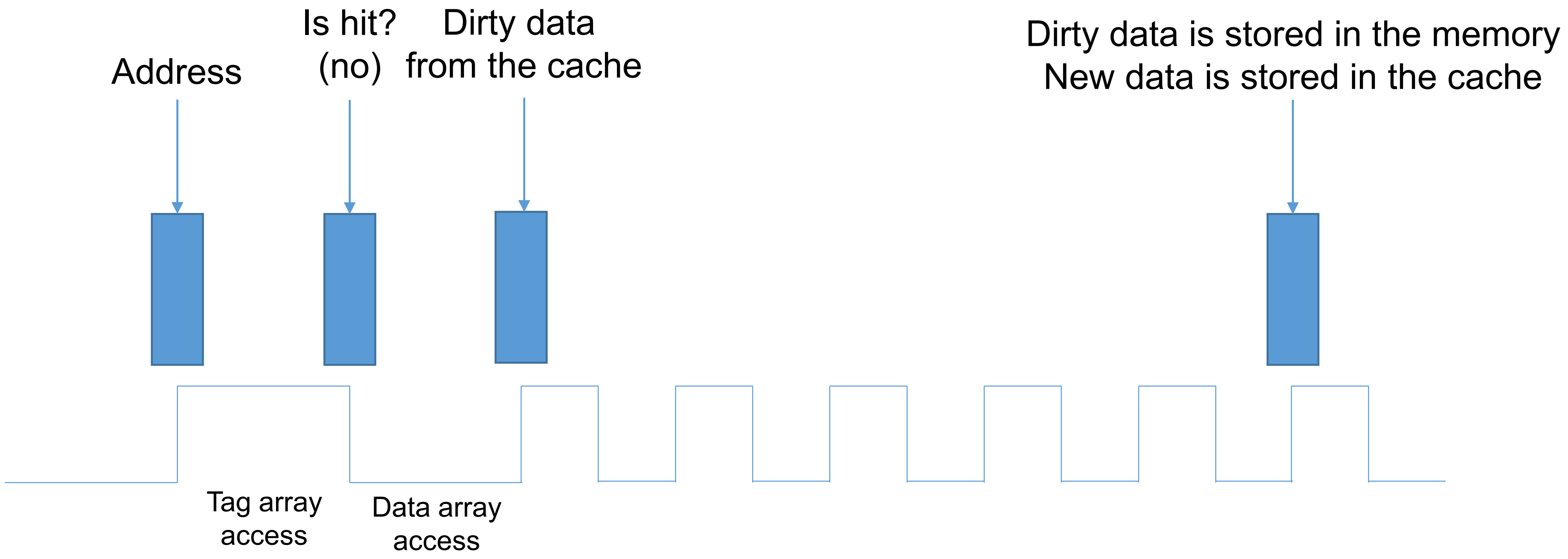
# Waveform (WRITE-THROUGH)

- Usually comes with write-no-allocate
  - Cache miss (WRITE request)
    - ✓ WRITE data into the memory



# Waveform (WRITE-BACK)

- Usually comes with write-allocate
  - Cache miss (WRITE request)
    - ✓ READ data from the memory + WRITE dirty data into the memory



# Lab Assignment 06 (4/5)

- Memory requirements
  - You can use either a 2-port RAM or a single-port RAM
  - Latencies of RAM
    - ✓ Should be serialized

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Single port | REQ 0 | BUSY | BUSY | BUSY | BUSY | BUSY | BUSY | REQ 1 | BUSY | ... |
| Dual port | REQ 0 | BUSY | BUSY | BUSY | BUSY | BUSY | BUSY | REQ 2 | BUSY | ... |
| | BUSY | BUSY | REQ1 | BUSY | BUSY | BUSY | BUSY | BUSY | BUSY | REQ3 |

⇒ Different ports can handle independent requests!

# Lab Assignment 06 (5/5)

- For the report, the following contents should be included
  - Describe your design choice
  - Calculate the hit (or miss) ratio
    - ✓ Hit ratio = (# of hits) / (# of memory accesses)
  - Compare the performance
    - ✓ CPU without cache vs. CPU with cache
    - ✓ You should use the new latencies!

# Announcement

- A skeleton code from this lab is not provided.
  You should implement cache *on your previous implementation of pipelined CPU*.

- You *cannot begin* this assignment unless you have finished previous work.