香港中文大學
The Chinese University of Hong Kong

CSCI2510 Computer Organization

Tutorial 09: Associative Mapping Implementation

Tsun-Yu YANG

yangty@cse.cuhk.edu.hk

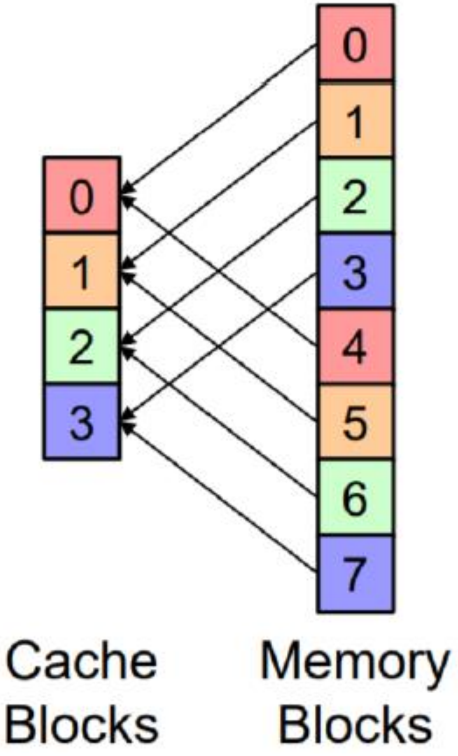
Outline

- Review of Associative Mapping
- Replacement Algorithms
  - LRU Replacement Algorithm
  - FIFO Replacement Algorithm
- Implementation of Associative Mapped Cache with MASM Code
- Hint: How to implement Set-Associative Mapped Cache with MASM Code?

Review of Associative Mapping

Direct

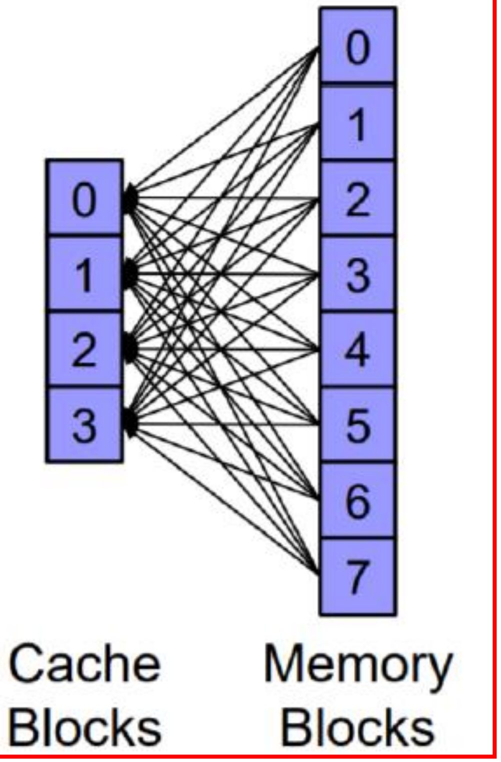
A Memory Block is **directly mapped** (%) to a Cache Block.



Associative

A Memory Block can be **mapped to any** Cache Block.

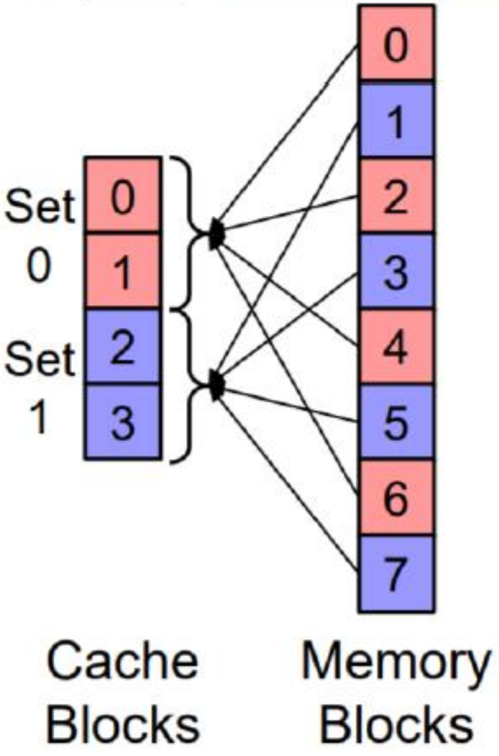
(First come first serve!)



Set Associative

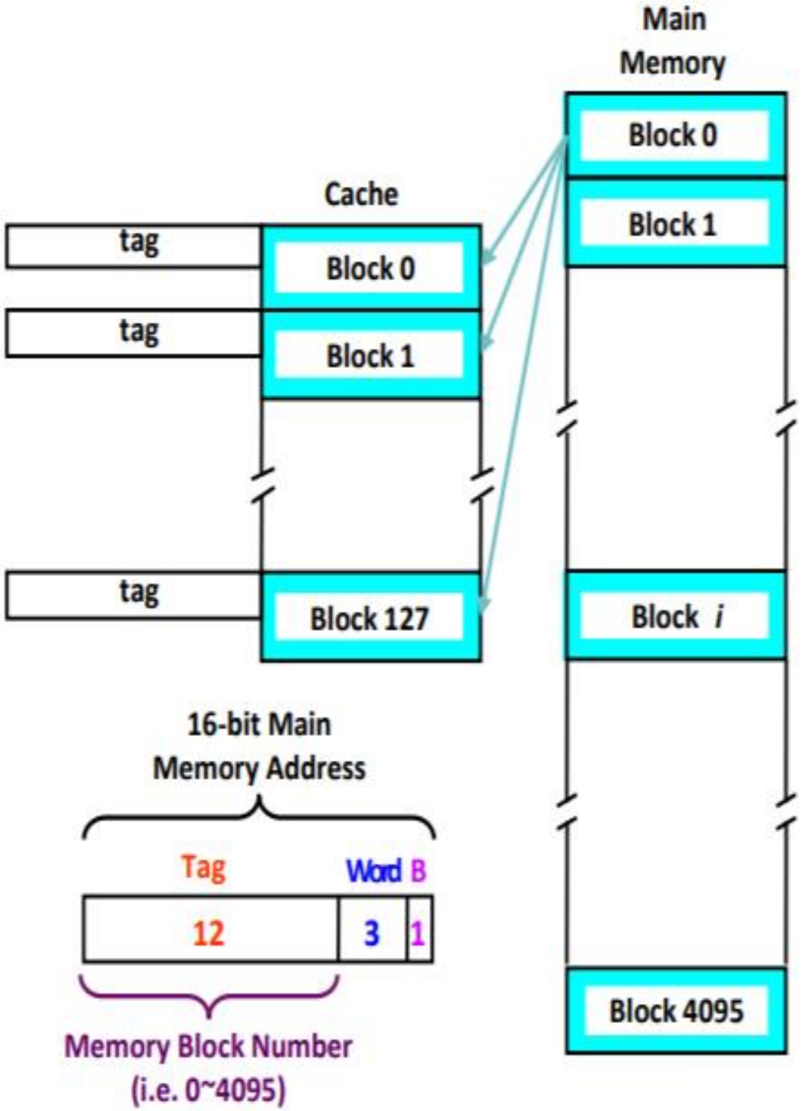
A Memory Block is **directly mapped** (%) to a **Cache Set**.

In a **Set**? **Associative**



Review of Associative Mapping

- **Direct Mapping:** A MB is restricted to a particular CB
  - MB #*j* → CB #(*j* mod 128)
- **Associative Mapping:** Allow a MB to be mapped to any CB in the cache
- In other words, it requires more space to store tag.



What if Cache Miss happened?

- Define: When the requested memory block is not in the Cache, we called it as a *Cache Miss*
  - Replace the old cache block with the new memory block

- The replacement for **Direct Mapping** is trivial:
  - The mapping between MB and CB is pre-determined
  - Simply replace the pre-determined CB with the new MB

- The replacement for **Associative** and **Set-Associative Mapping** is non-trivial:
  - Because we have the flexibility to select a cache block to be replaced → so we need replacement algorithm

Replacement Algorithms

- Optimal Replacement:
  - Always keep CBs, which will be used sooner, in the cache. (Impossible to know the future)
- Random Replacement:
  - Randomly replace a block (Easy to implement, but incur more Cache Miss)
- Least Recently Used (LRU) Replacement:
  - Replace the block that has gone the longest time without being accessed (Based on temporal locality)
- First-In-First-Out (FIFO) Replacement:
  - The first cached memory block will also be the first one to be replaced.

Example of LRU Algorithm

- Suppose we have a program that make the following access pattern:

Access Address	X[0]	X[1]	X[2]	X[1]	X[0]
Time Stamp	1	2	3	4	5

- Also we have a cache with 2 cache blocks:

	CB0	CB1
Cache:	-1	-1
Time Stamp:	-1	-1

Always find the smallest time stamp block!

Example of LRU Algorithm

- Suppose we have a program that make the following access pattern:

Access Address	X[0]	X[1]	X[2]	X[1]	X[0]
Time Stamp	1	2	3	4	5

- Also we have a cache with 2 cache blocks:

	CB0	CB1
Cache:	-1→X[0]	-1
Time Stamp:	-1→1	-1

Example of LRU Algorithm

- Suppose we have a program that make the following access pattern:

Access Address	X[0]	X[1]	X[2]	X[1]	X[0]
Time Stamp	1	2	3	4	5

- Also we have a cache with 2 cache blocks:

	CB0	CB1
Cache:	X[0]	-1→X[1]
Time Stamp:	1	-1→2

Example of LRU Algorithm

- Suppose we have a program that make the following access pattern:

Access Address	X[0]	X[1]	X[2]	X[1]	X[0]
Time Stamp	1	2	3	4	5

- Also we have a cache with 2 cache blocks:

	CB0	CB1
Cache:	X[0]→X[2]	X[1]
Time Stamp:	1→3	2

Example of LRU Algorithm

- Suppose we have a program that make the following access pattern:

Access Address	X[0]	X[1]	X[2]	X[1]	X[0]
Time Stamp	1	2	3	4	5

- Also we have a cache with 2 cache blocks:

	CB0	CB1
Cache:	X[2]	X[1]
Time Stamp:	3	2→4

Example of LRU Algorithm

- Suppose we have a program that make the following access pattern:

Access Address	X[0]	X[1]	X[2]	X[1]	X[0]
Time Stamp	1	2	3	4	5

- Also we have a cache with 2 cache blocks:

	CB0	CB1
Cache:	X[2]→X[0]	X[1]
Time Stamp:	3→5	4

Example of FIFO Algorithm

- FIFO is similar with the LRU, but it doesn’t update the Time Stamp when there is a *Cache Hit*.
- Let’s assume the same conditions:

Access Address	X[0]	X[1]	X[2]	X[1]	X[0]
Time Stamp	1	2	3	4	5

	CB0	CB1
Cache:	-1	-1
Time Stamp:	-1	-1

Example of FIFO Algorithm

- FIFO is similar with the LRU, but it doesn’t update the Time Stamp when there is a *Cache Hit*.
- Let’s assume the same conditions:

Access Address	X[0]	X[1]	X[2]	X[1]	X[0]
Time Stamp	1	2	3	4	5

	CB0	CB1
Cache:	-1→X[0]	-1
Time Stamp:	-1→1	-1

Example of FIFO Algorithm

- FIFO is similar with the LRU, but it doesn’t update the Time Stamp when there is a *Cache Hit*.
- Let’s assume the same conditions:

Access Address	X[0]	X[1]	X[2]	X[1]	X[0]
Time Stamp	1	2	3	4	5

	CB0	CB1
Cache:	X[0]	-1→X[1]
Time Stamp:	1	-1→2

Example of FIFO Algorithm

- FIFO is similar with the LRU, but it doesn’t update the Time Stamp when there is a *Cache Hit*.
- Let’s assume the same conditions:

Access Address	X[0]	X[1]	X[2]	X[1]	X[0]
Time Stamp	1	2	3	4	5

	CB0	CB1
Cache:	X[0]→X[2]	X[1]
Time Stamp:	1→3	2

Example of FIFO Algorithm

- FIFO is similar with the LRU, but it doesn’t update the Time Stamp when there is a *Cache Hit*.
- Let’s assume the same conditions:

Access Address	X[0]	X[1]	X[2]	X[1]	X[0]
Time Stamp	1	2	3	4	5

	CB0	CB1
Cache:	X[2]	X[1]
Time Stamp:	3	2

Example of FIFO Algorithm

- FIFO is similar with the LRU, but it doesn’t update the Time Stamp when there is a *Cache Hit*.
- Let’s assume the same conditions:

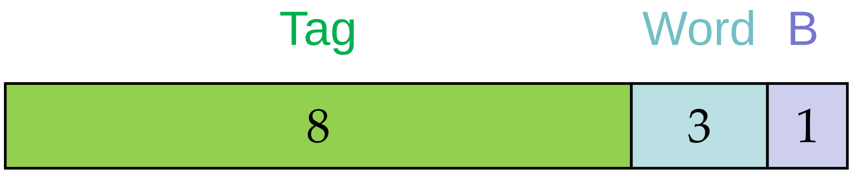
Access Address	X[0]	X[1]	X[2]	X[1]	X[0]
Time Stamp	1	2	3	4	5

	CB0	CB1
Cache:	X[2]	X[1]→X[0]
Time Stamp:	3	2→5

Implementation of Associative Mapping

- In this example, we have the following configuration:
  - Memory address is 16-bit
  - Totally 4 cache blocks (for the ease of demonstration), which every block has 8 words and each word is 2 bytes

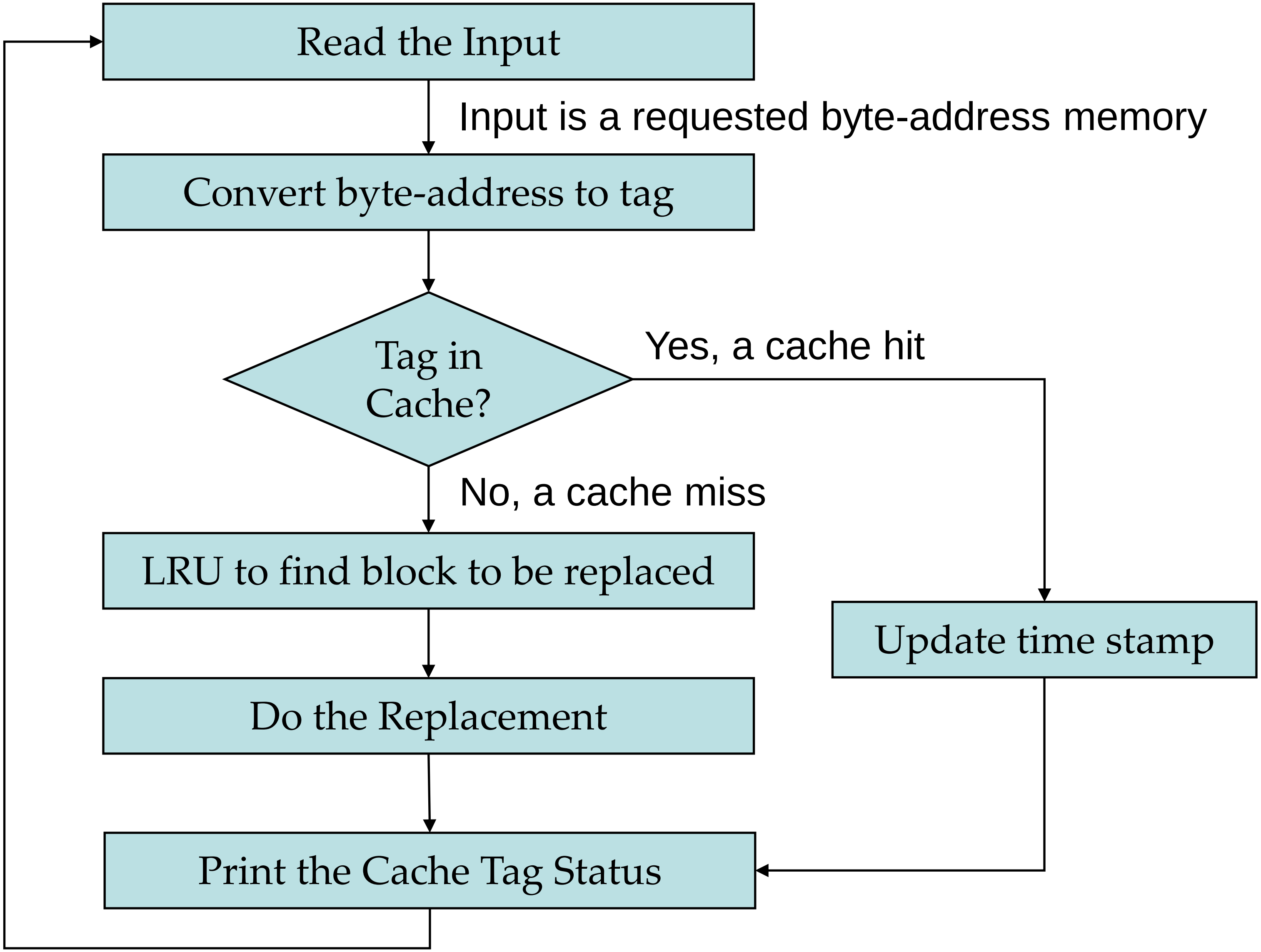
Our implementation:



Tag	Word	B
8	3	1

- **Please note that since we cannot directly manage CPU cache, we allocate a memory space to simulate cache for the sake of practice**

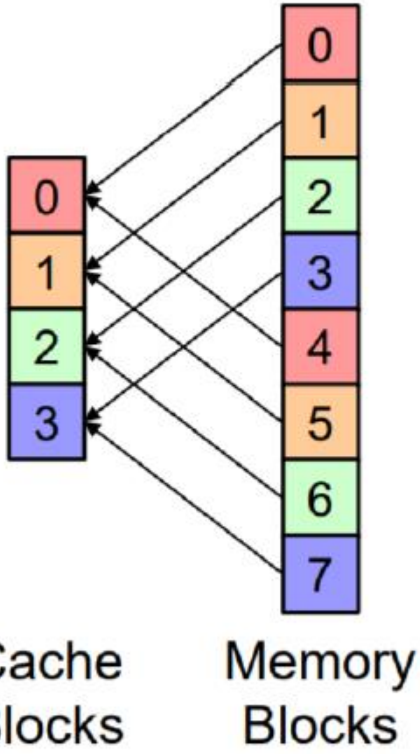
Flowchart of Associative Cache


Read the Input
Input is a requested byte-address memory
Convert byte-address to tag
Tag in Cache?
Yes, a cache hit
No, a cache miss
LRU to find block to be replaced
Update time stamp
Do the Replacement
Print the Cache Tag Status

How to implement Set-Associative?

Direct

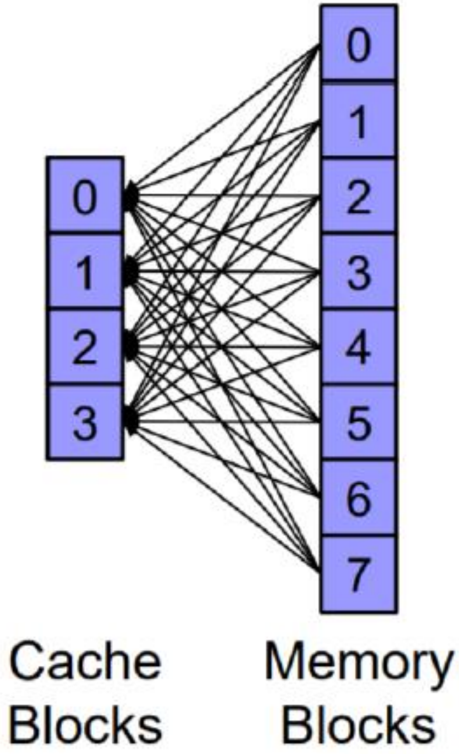
A Memory Block is **directly mapped** (%) to a Cache Block.



Associative

A Memory Block can be **mapped to any** Cache Block.

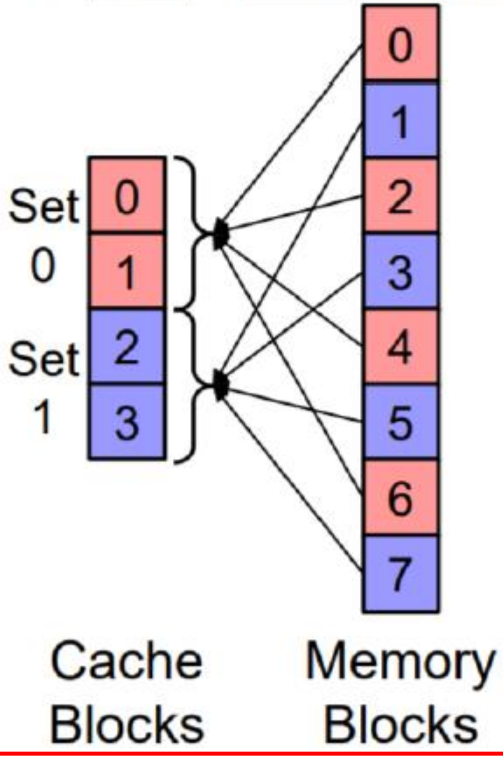
(First come first serve!)



Set Associative

A Memory Block is **directly mapped** (%) to a **Cache Set**.

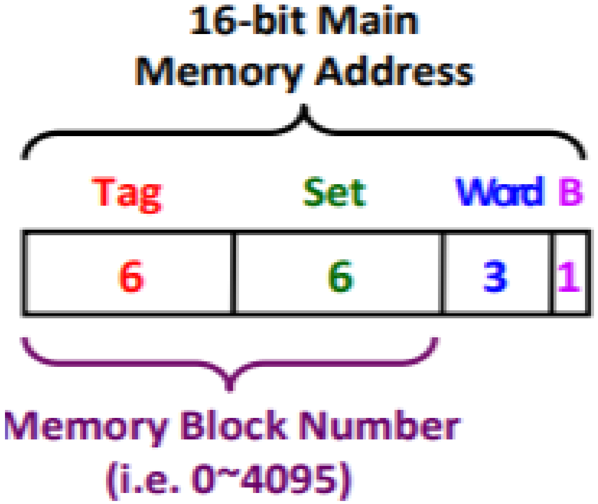
In a **Set**? **Associative**!



How to implement Set-Associative?

- Set-Associative Mapping is the combination of the Direct Mapping and Associative Mapping
  - Use Direct Mapping technique to find the corresponding Set ID
  - Use Associative Mapping technique to deal with tag searching and replacement

- Important Notes:
  1. Find the Set ID
  2. Get the Tag value
  3. Search the CBs in Set w/ Tag value
  4. If Cache miss, do replacement algorithm within the Set



Summary

- Review of Associative Mapping
- Replacement Algorithms
  - LRU Replacement Algorithm
  - FIFO Replacement Algorithm
- Implementation of Associative Mapped Cache with MASM Code
- Hint: How to implement Set-Associative Mapped Cache with MASM Code?