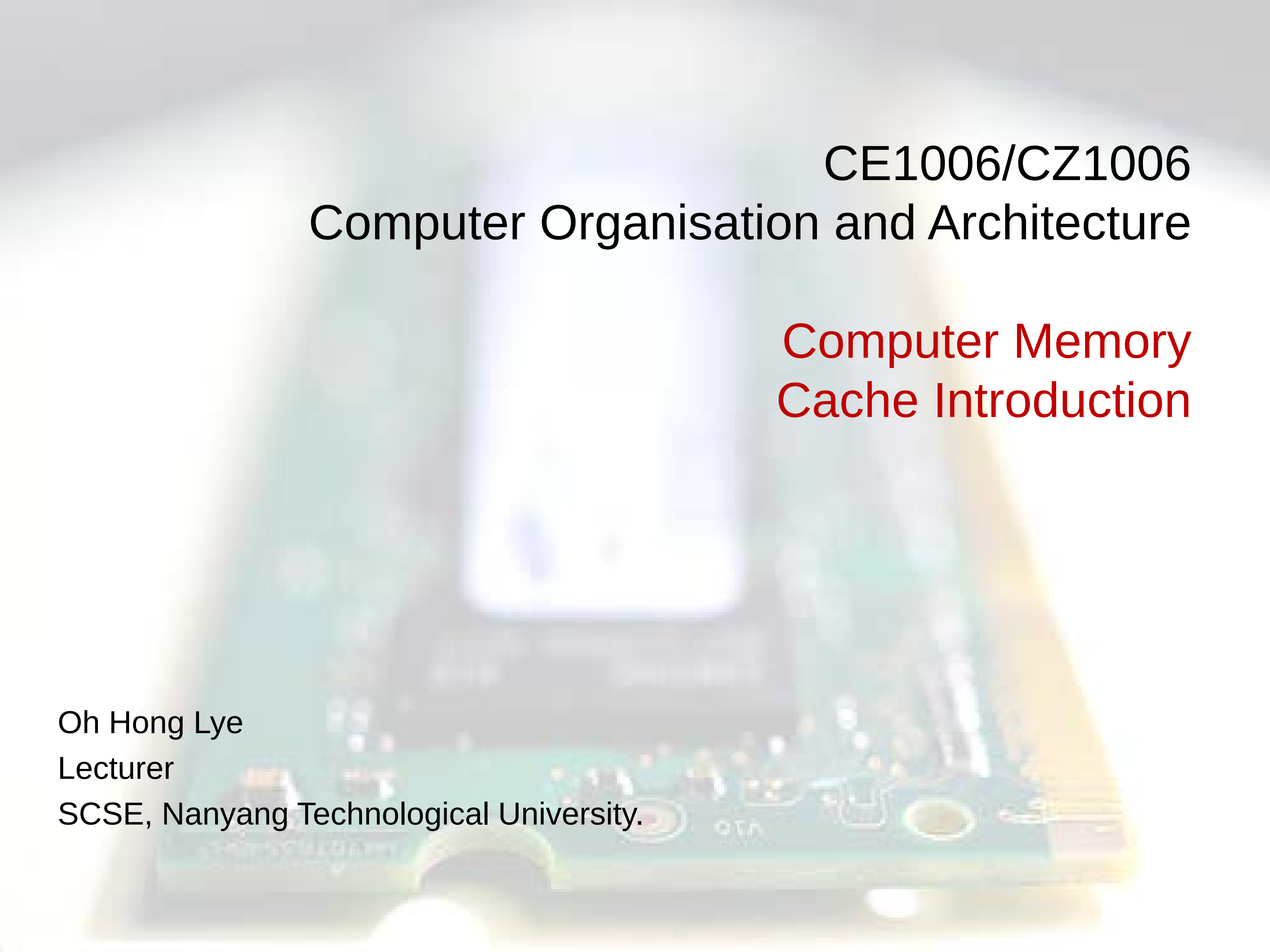
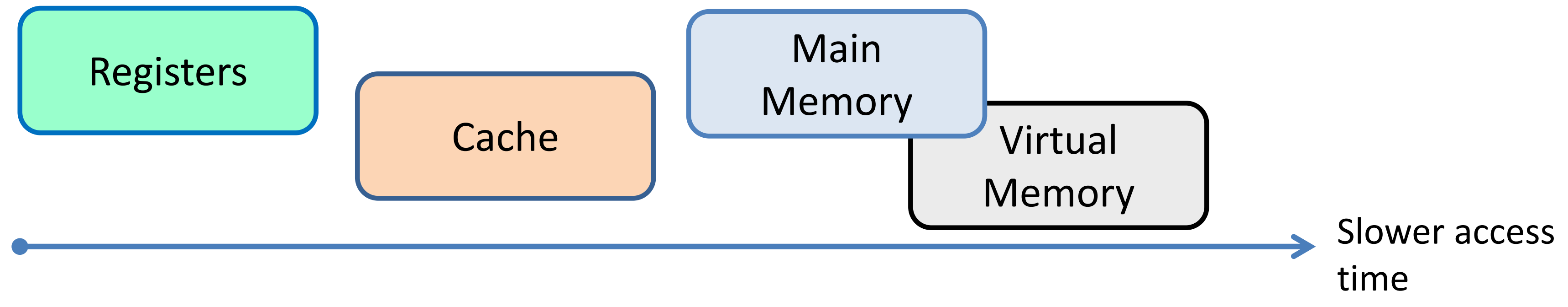


CE1006/CZ1006
Computer Organisation and Architecture

Computer Memory
Cache Introduction

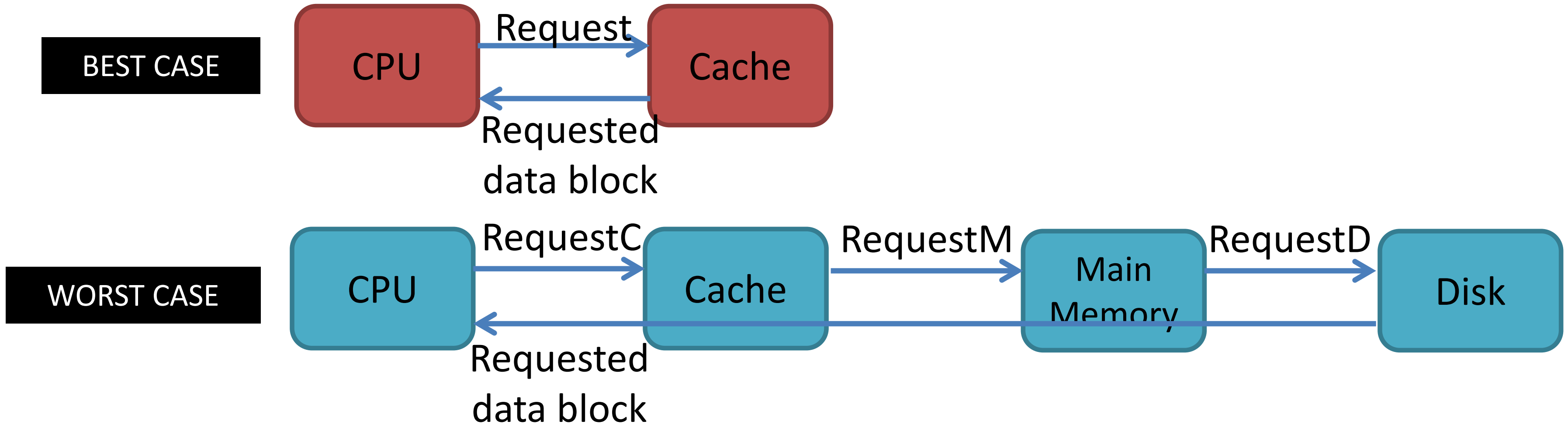
Oh Hong Lye
Lecturer
SCSE, Nanyang Technological University.

System Memory



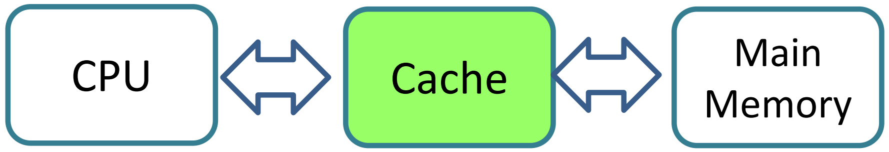
- Review of Memory Hierarchy
- Registers and caches typically available on the processor.
  - Implemented with fast memory and resides on the same chip.
  - Main reason why access time is shorter.
- Virtual memory helps extend the address space of the system.
- Tradeoff
  - More cache memory on-chip is expensive.
  - More virtual memory may require longer access time as copying of required data to Main Memory may be needed.

CPU Memory Access



- To access a particular piece of data, the CPU first sends a request to its nearest memory, usually cache.
- If the data is not in cache, a query is then sent to the main memory.
- If the data is not in main memory, then the request goes to disk.
- Once the data is located, the **required data and a number of its nearby data elements** are fetched into cache memory simultaneously.

Cache



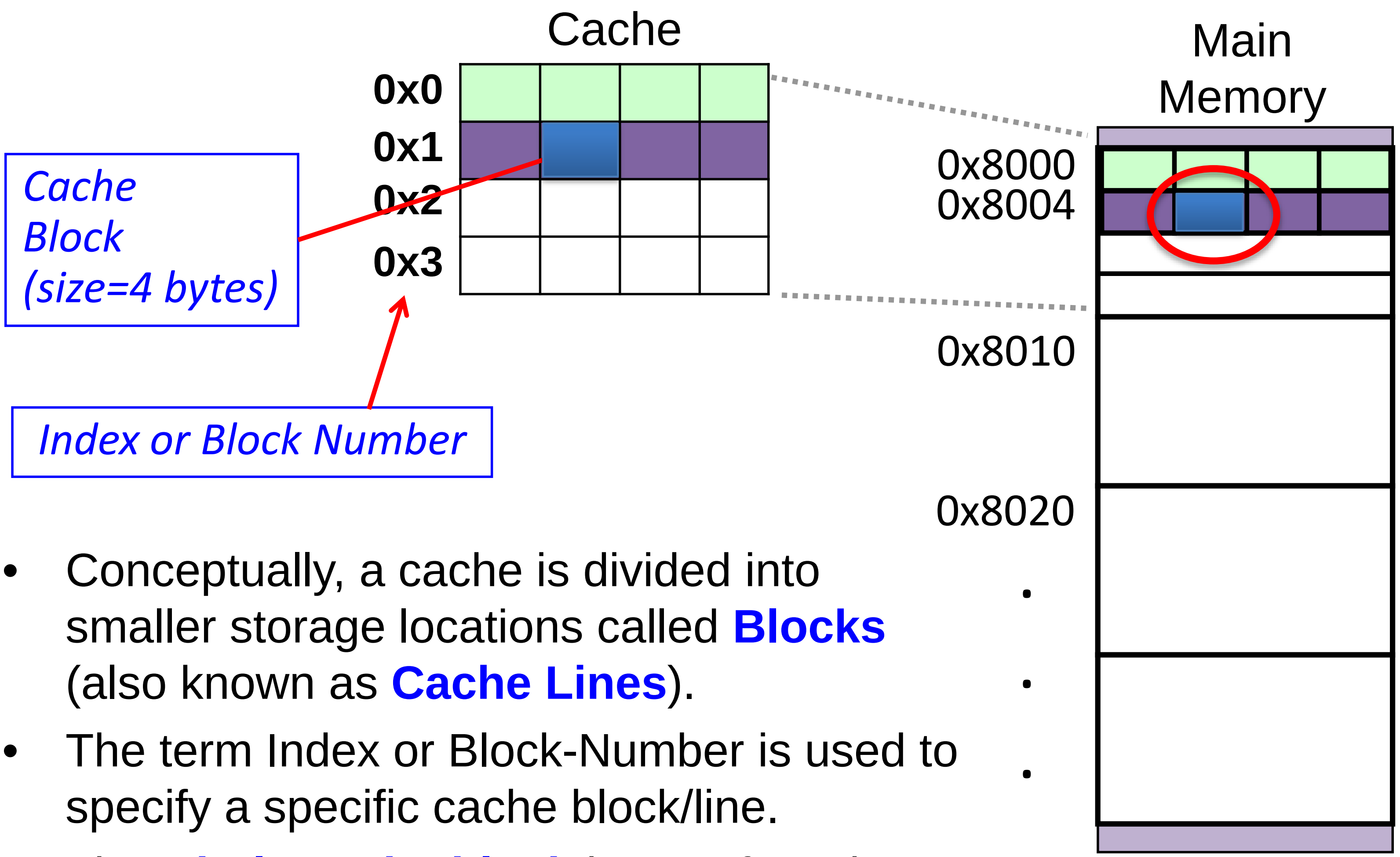
- Issue: CPU speed is typically much **faster** than external memory.
  - CPU speed in Ghz region
  - External Memory Speed in 100s of Mhz region.
- Need a fast memory to act as a **buffer** between the Main memory and CPU.
- The purpose of cache memory is to speed up accesses by storing/fetching **recently used data** closer to the CPU instead of the main memory (slower access).
- Potentially able to improve the overall system performance drastically.

CE1006/CZ1006 Computer Organisation and Architecture

Computer Memory
Cache Mapping Scheme
Direct Mapped Cache

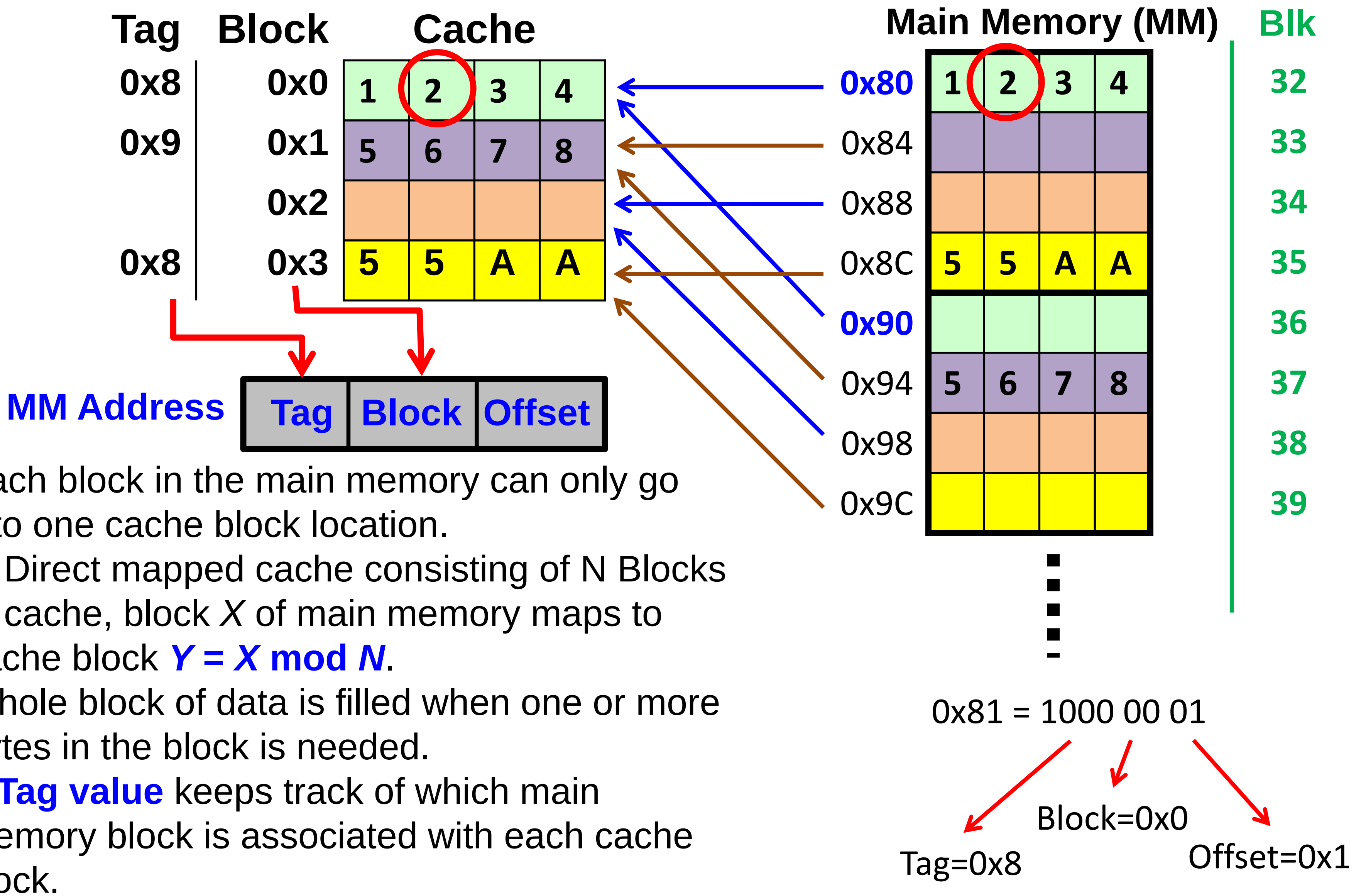
Oh Hong Lye
Lecturer
SCSE, Nanyang Technological University.

Terms used in Cache Mapping

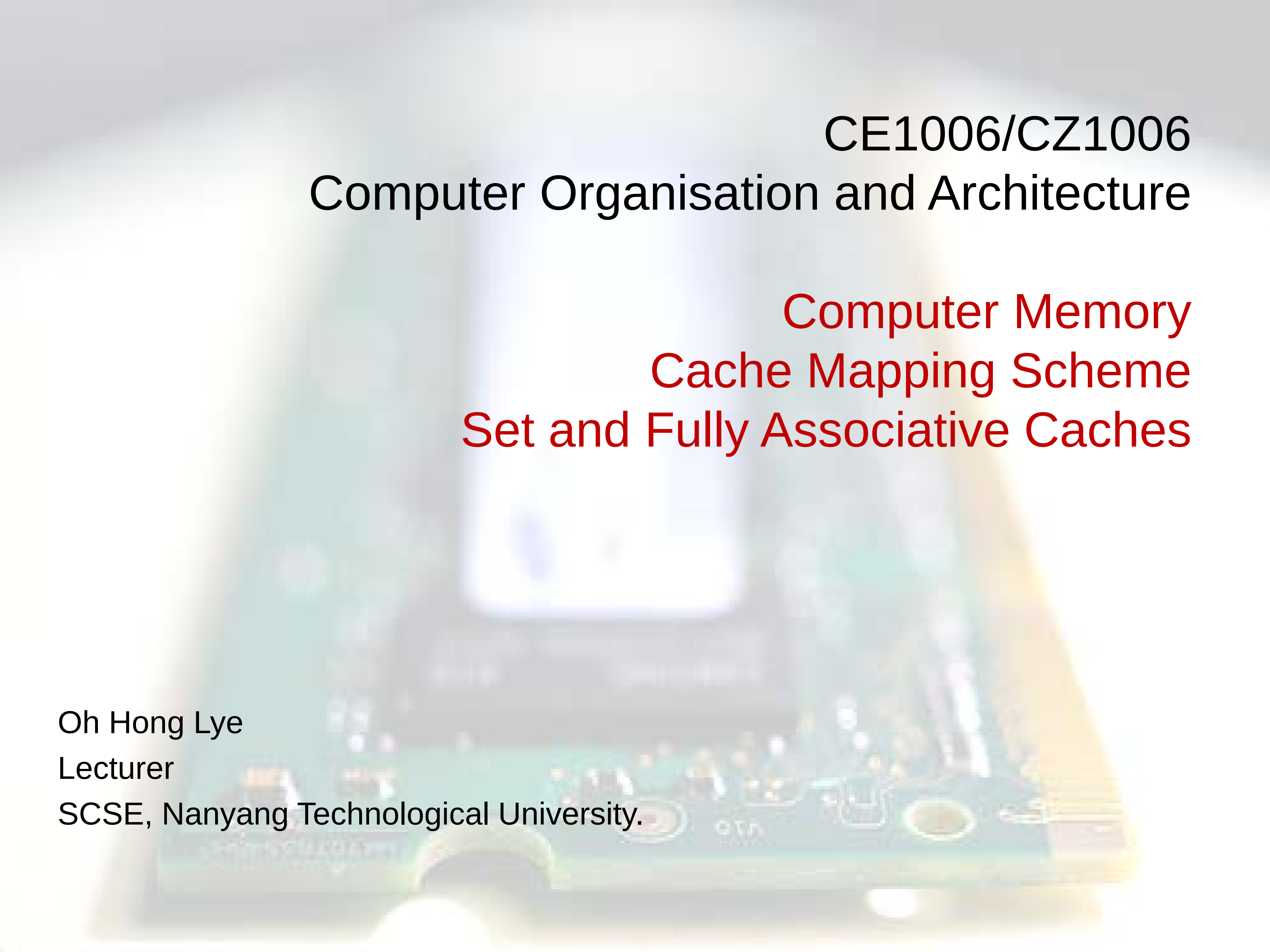


- Conceptually, a cache is divided into smaller storage locations called **Blocks** (also known as **Cache Lines**).
- The term Index or Block-Number is used to specify a specific cache block/line.
- The **whole cache block** is transferred when one or more bytes in the block is needed.

Direct Mapped Cache



- Each block in the main memory can only go into one cache block location.
- In Direct mapped cache consisting of N Blocks of cache, block *X* of main memory maps to cache block ***Y* = *X* mod *N***.
- Whole block of data is filled when one or more bytes in the block is needed.
- A **Tag value** keeps track of which main memory block is associated with each cache block.



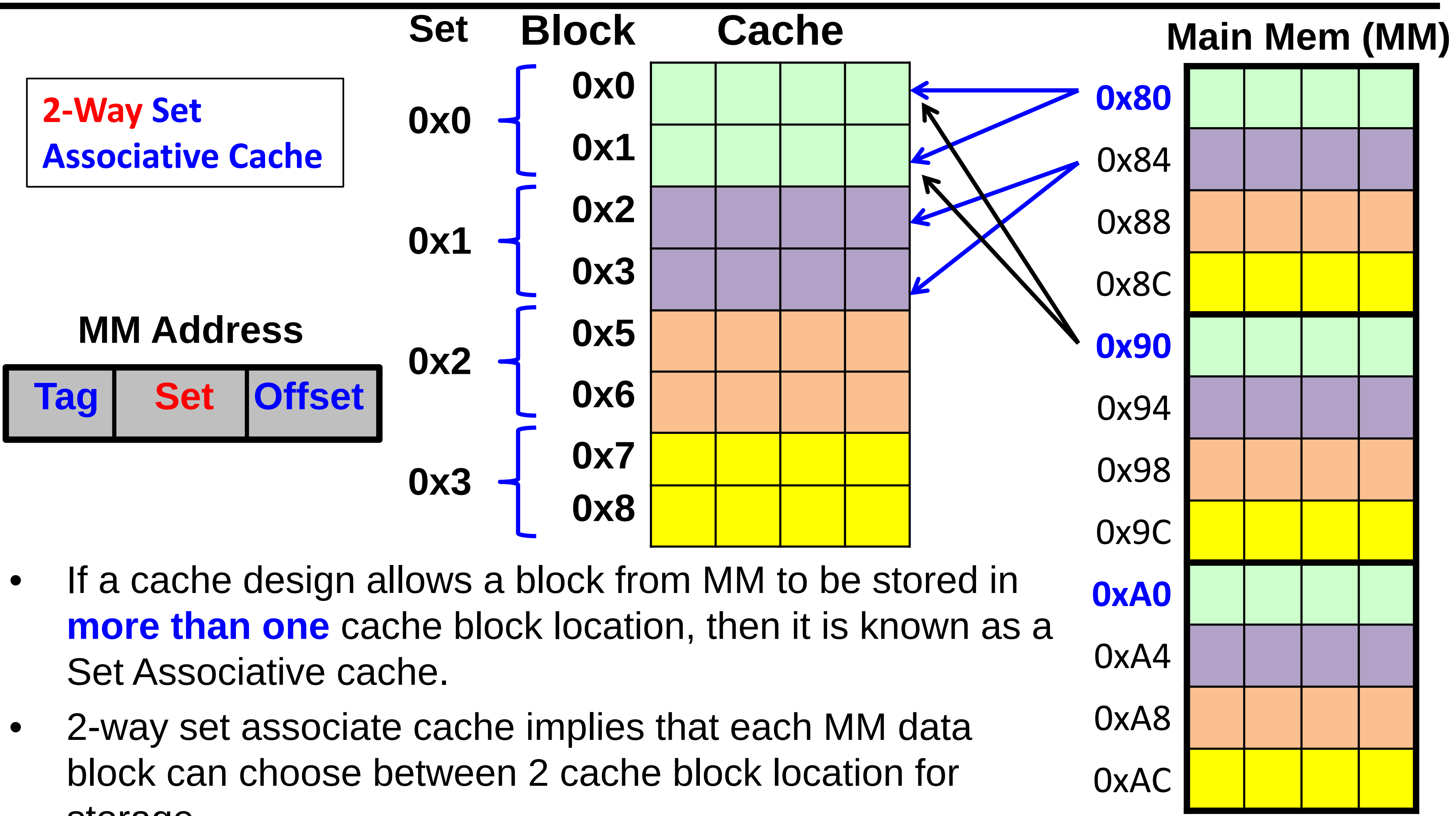
CE1006/CZ1006
Computer Organisation and Architecture

Computer Memory
Cache Mapping Scheme
Set and Fully Associative Caches

Oh Hong Lye
Lecturer
SCSE, Nanyang Technological University.

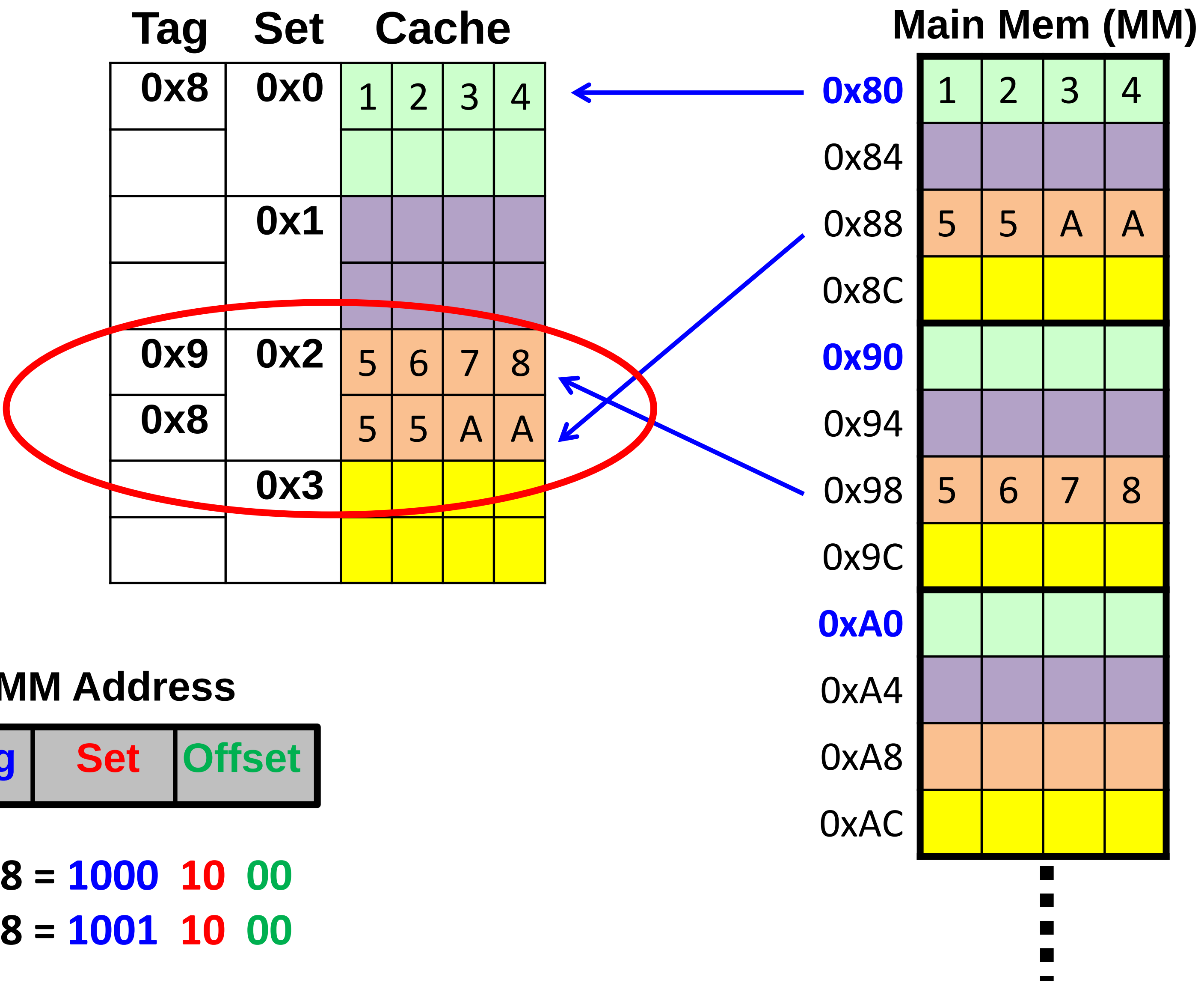
N-way Set Associative Cache

2-Way Set Associative Cache



- If a cache design allows a block from MM to be stored in **more than one** cache block location, then it is known as a Set Associative cache.
- 2-way set associate cache implies that each MM data block can choose between 2 cache block location for storage.
- Direct Mapped Cache can be viewed as 1-way set associate cache.

2-Way Set Associative Cache



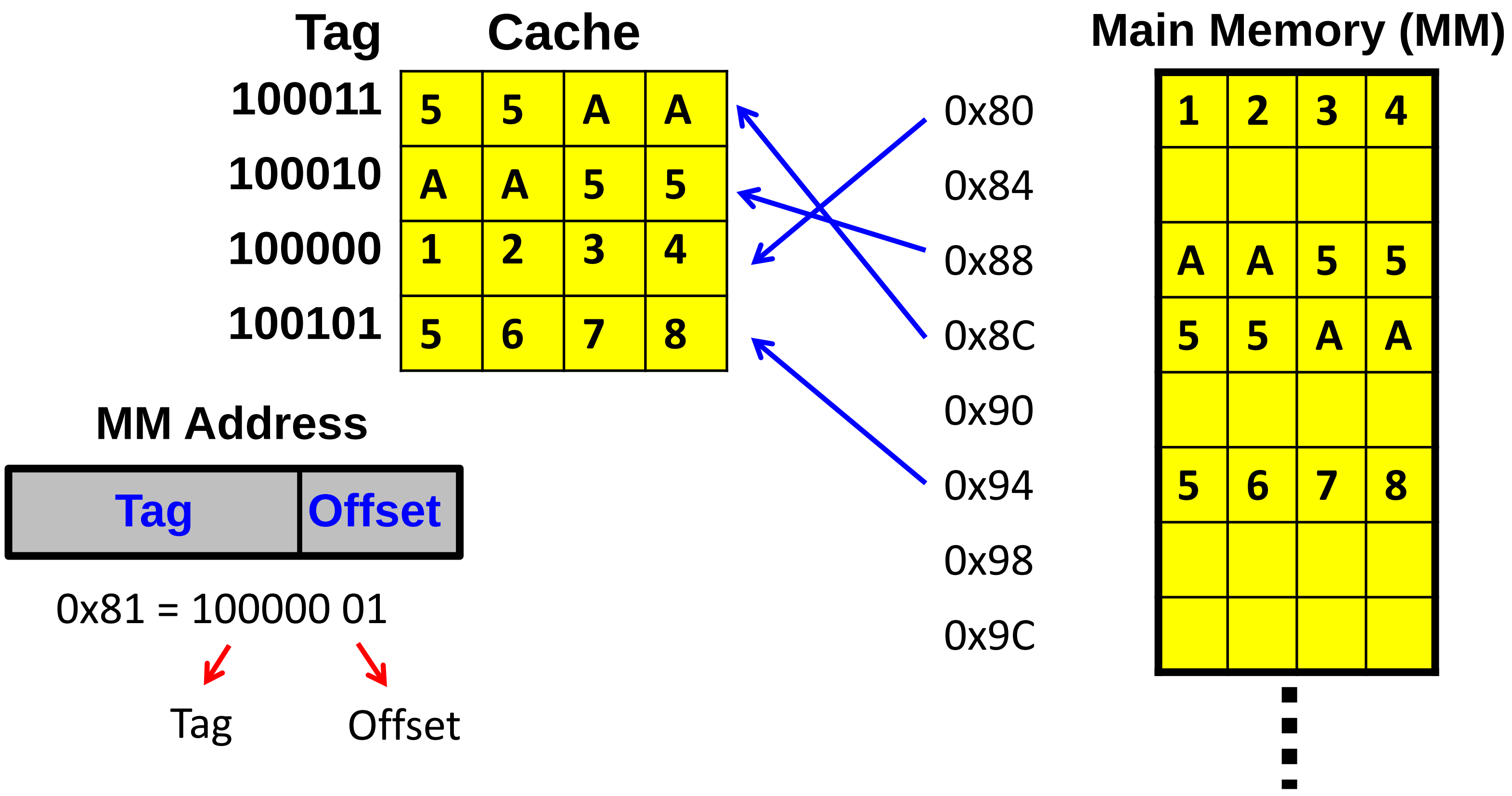
MM Address

Tag	Set	Offset

0x88 = 1000 10 00

0x98 = 1001 10 00

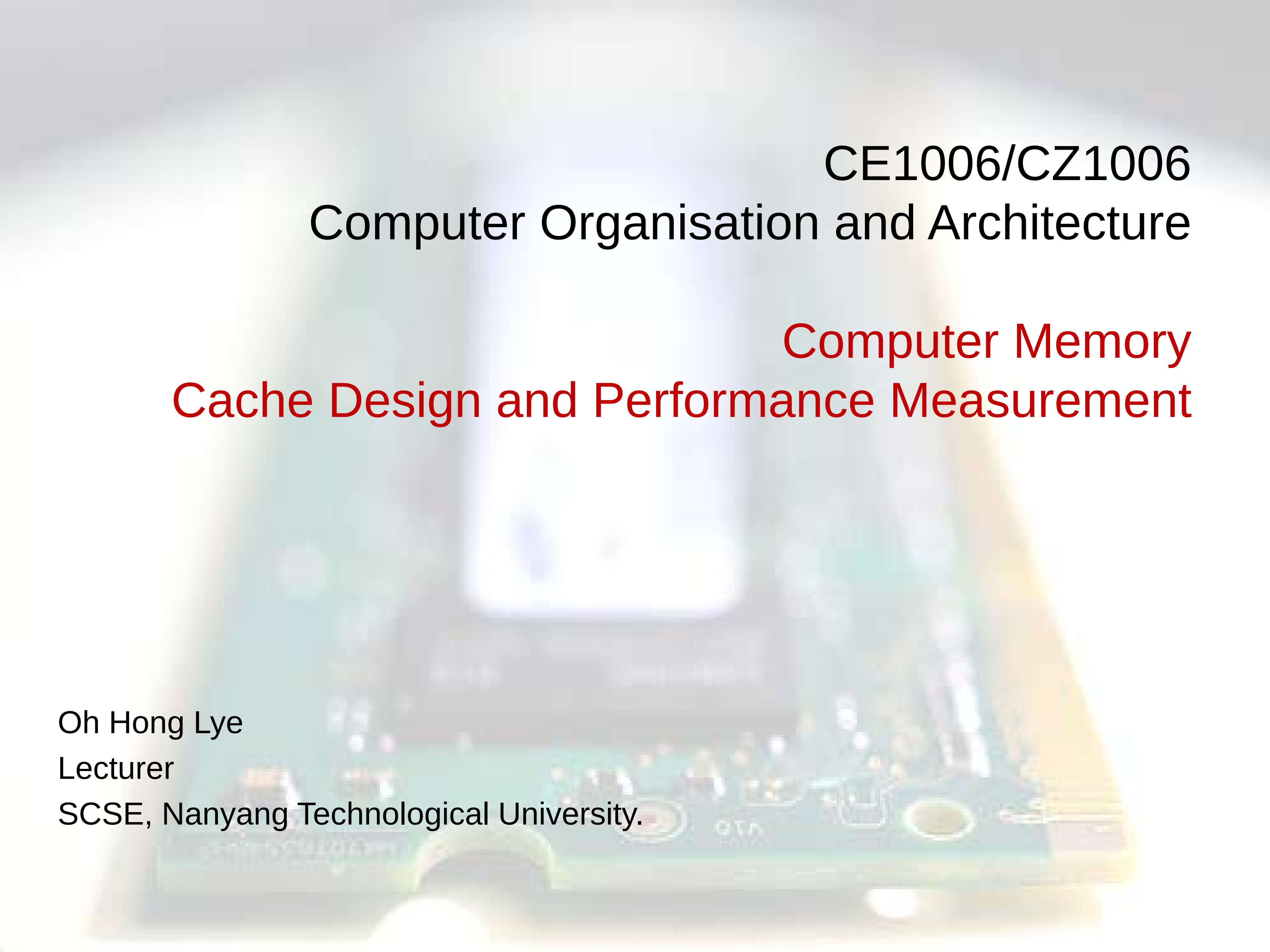
Fully Associative Cache



- Data block from main memory can be placed in **ANY** cache block.
- Issue
  - Which cache block to use for storage?
  - If cache is full, Which cache block to evict first?

Comparison of Mapping Scheme

- Direct Mapped Cache
  - Advantage
    - Simple to implement
    - Items can be found in the Cache easily and rapidly
  - Disadvantage
    - Very prone to Cache thrashing
    - Hit rate lowest among the three scheme.
- N-Way Set Associative Cache
  - Advantage
    - Less Cache Thrashing
    - Higher Hit Rate for the same Cache size
  - Disadvantage
    - More complex design
    - Longer search time
- Fully Associative
  - Extreme case of N-Way Set Associative Cache.



CE1006/CZ1006
Computer Organisation and Architecture

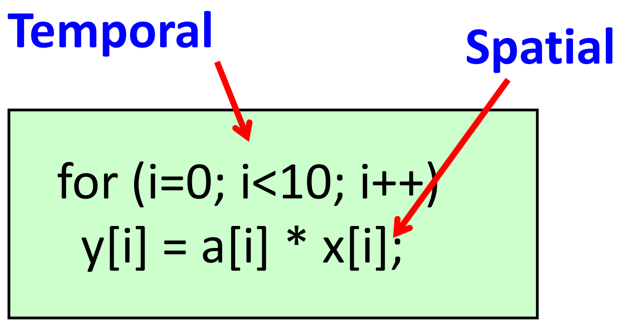
Computer Memory
Cache Design and Performance Measurement

Oh Hong Lye
Lecturer
SCSE, Nanyang Technological University.

Principles of Cache Design

- Two distinct tasks need to be performed by the Cache
- **Storage**
  - Decide what will be stored and replaced in the Cache
    - Principle of Locality
- **Retrieval**
  - Check if an item is stored in the Cache and retrieve it efficiently
  - Mapping Scheme
    - Direct Mapped
    - Fully Associative
    - Set Associative

Cache Design – Storage



```
for (i=0; i<10; i++)
  y[i] = a[i] * x[i];
```

- The decision of what to store is the most fundamental question in Cache architecture design
- All Caches use the notion of localities
- The **principle of locality** tells us that once a byte is accessed, it is likely a **nearby data element** will be needed soon.
- Hence, as you have seen in the cache mapping schemes, an **entire block** of data is copied after a hit.

Principle of locality

- Locality of **Space** (or Spatial Locality)
  - **Code/Data that is close together** is likely to be accessed together
  - Burst Cache fills (Memory Reads with more than one word at a time) is preferable due to spatial locality.

- Locality of **Time** (or Temporal Locality)
  - **Recently executed code** is more likely to be executed again
  - Used to decide which item to replace in the Cache

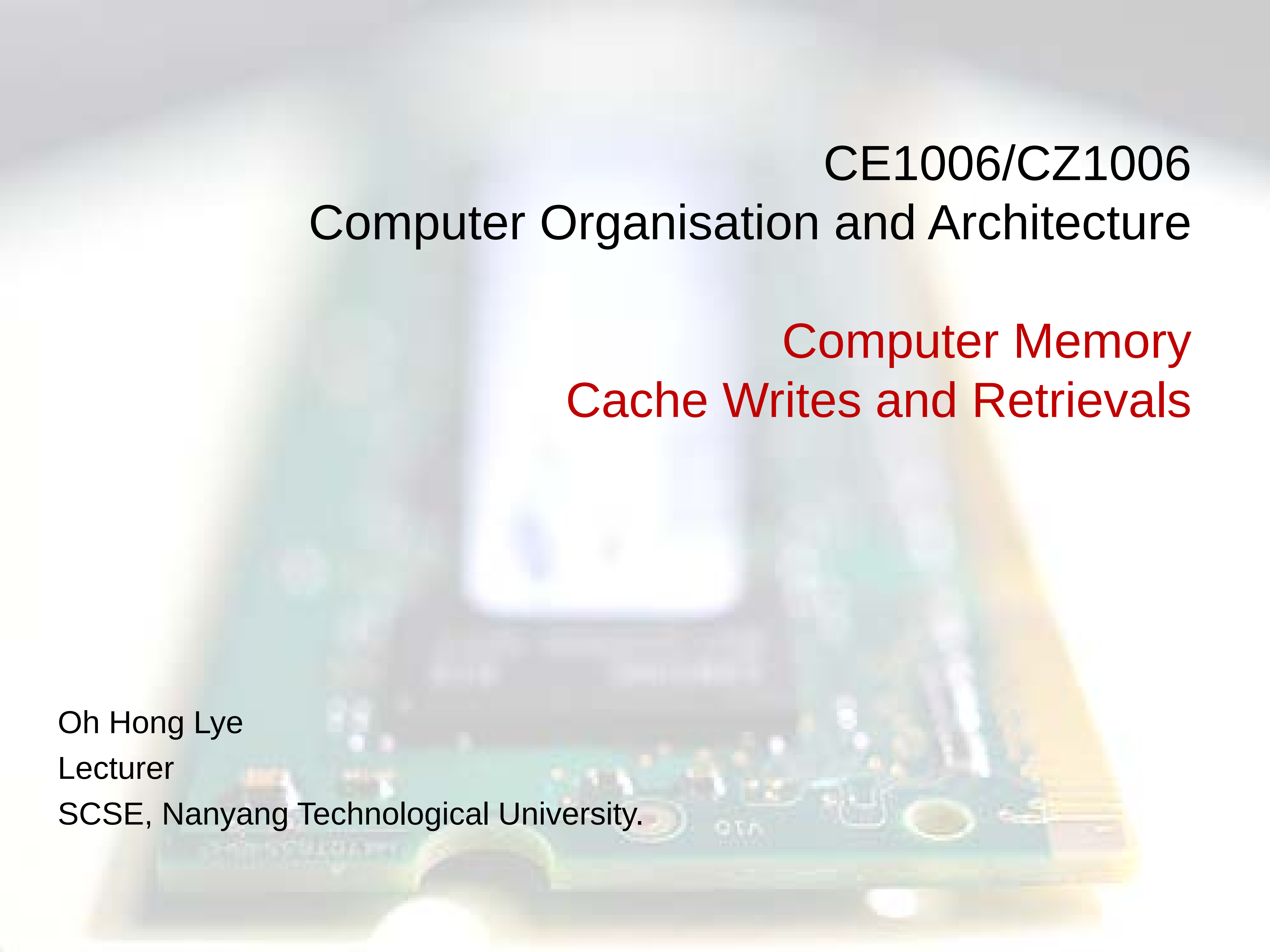
- Cache performance depends upon programs exhibiting good locality.
  - Some object-oriented programs have poor locality owing to their complex, dynamic structures.
  - Multi-Dimension Arrays should be accessed row/column wise depending on how the compiler stores the array in the memory.

Cache Memory Replacement Policy

- A **least recently used (LRU)** algorithm keeps track of the last time that a block was assessed and evicts the block that has been unused for the longest period of time. The disadvantage of this approach is its complexity: LRU has to maintain an access history for each block, which ultimately slows down the cache.

- **First-in, first-out (FIFO)** is a popular cache replacement policy. In FIFO, the block that has been in the cache the longest, regardless of when it was last used.

- A **random** replacement policy does what its name implies: It picks a block at random and replaces it with a new block. Random replacement can certainly evict a block that will be needed often or needed soon, but it never thrashes.



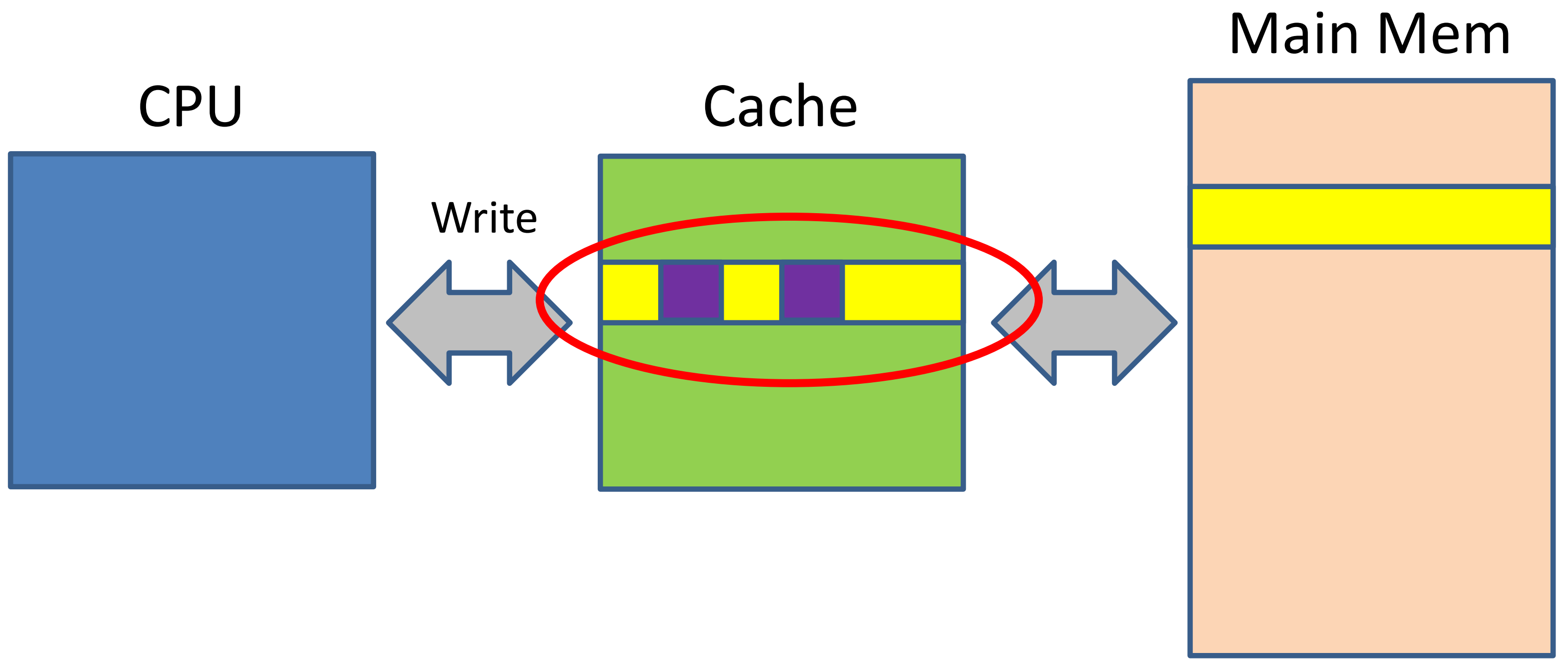
CE1006/CZ1006 Computer Organisation and Architecture

Computer Memory Cache Writes and Retrievals

Oh Hong Lye
Lecturer
SCSE, Nanyang Technological University.

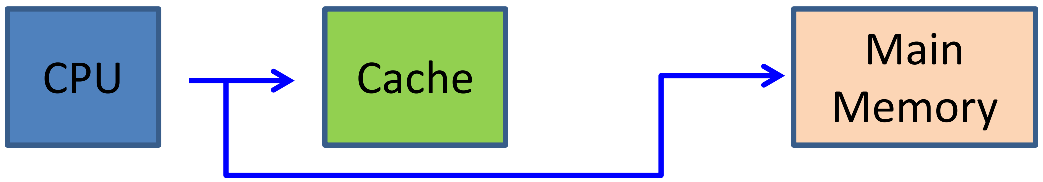
Dirty Block and Write Policy

- Cache replacement policies must take into account **dirty blocks**, those blocks that have been updated while they were in the cache.
- Dirty blocks must be written back to memory. A **write policy** determines how this will be done.
- There are two types of write policies; **write through** and **write back**.

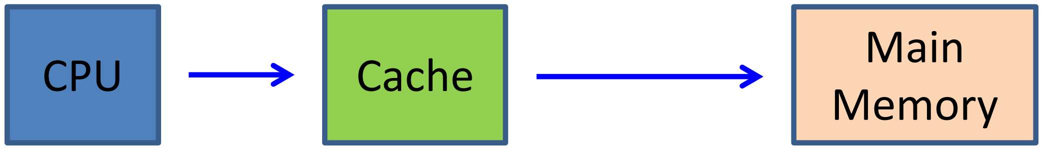


Cache Write Policy

- **Write through** updates cache and main memory **simultaneously** on every write.



- **Write back** (also called copyback) updates memory **only** when the block is selected for replacement.

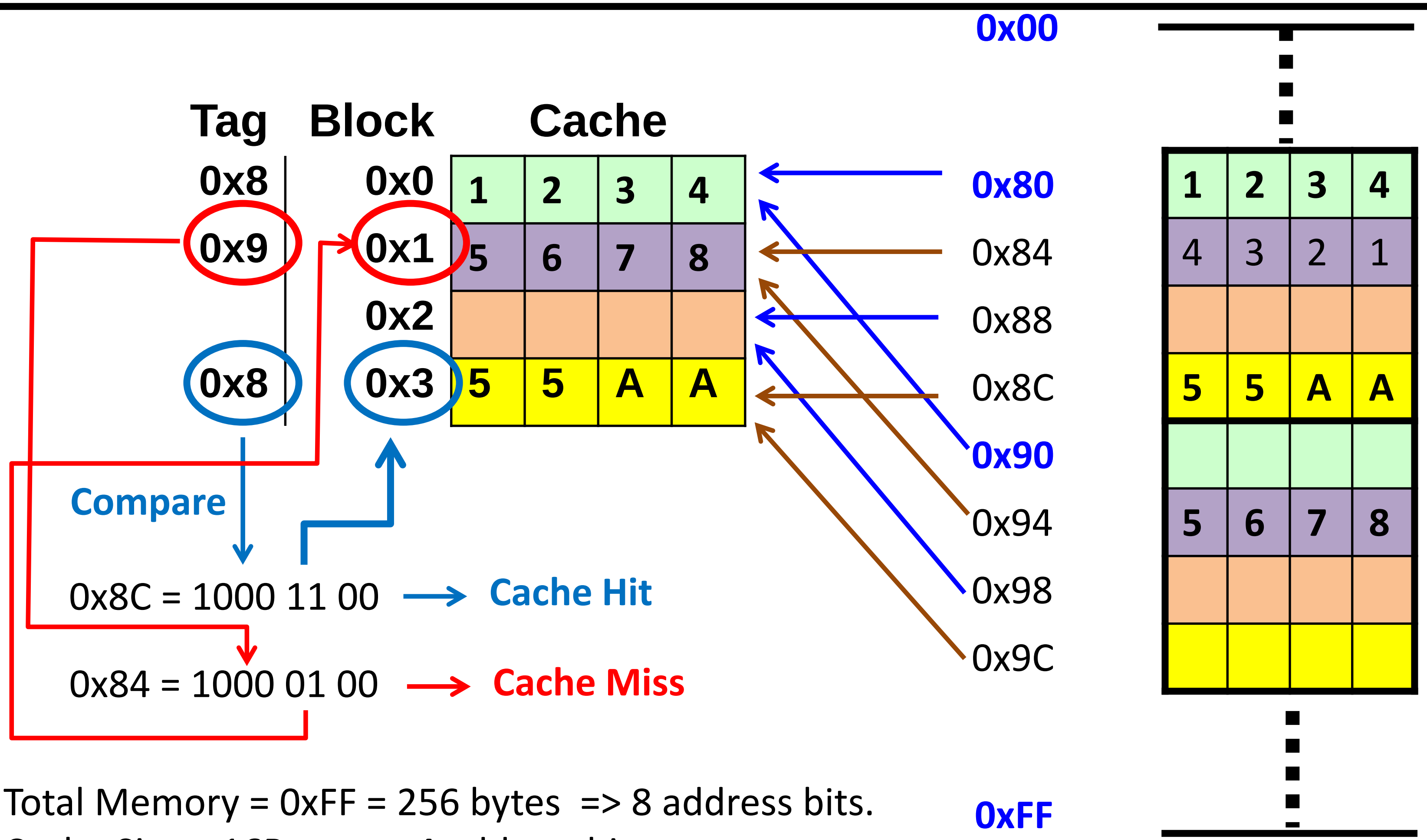


- **Write Miss** Policy
  - **Write allocate** => fetch on write. A write miss will cause the data block at the write-miss address to be **loaded to the cache**.
  - **Write-no-allocate** => A write miss will not cause the data block to be loaded to the cache. Data **Write** is done **directly** to its location in **main memory**.

Cache Design - Retrieval

- The time to find and retrieve and item from the Cache greatly influences the Cache performance
- A simple and fast algorithm is needed to store items in the Cache so that they can be quickly found
- This algorithm must make use of the Locality properties
- Illustrated in the mapping scheme discussed earlier.

Data Retrieval Example (Direct-Mapped Cache)



Total Memory = 0xFF = 256 bytes => 8 address bits.

Cache Size = 16Bytes => 4 address bits.

Cache Block Size = 4 Bytes => 2 address bits for offset.

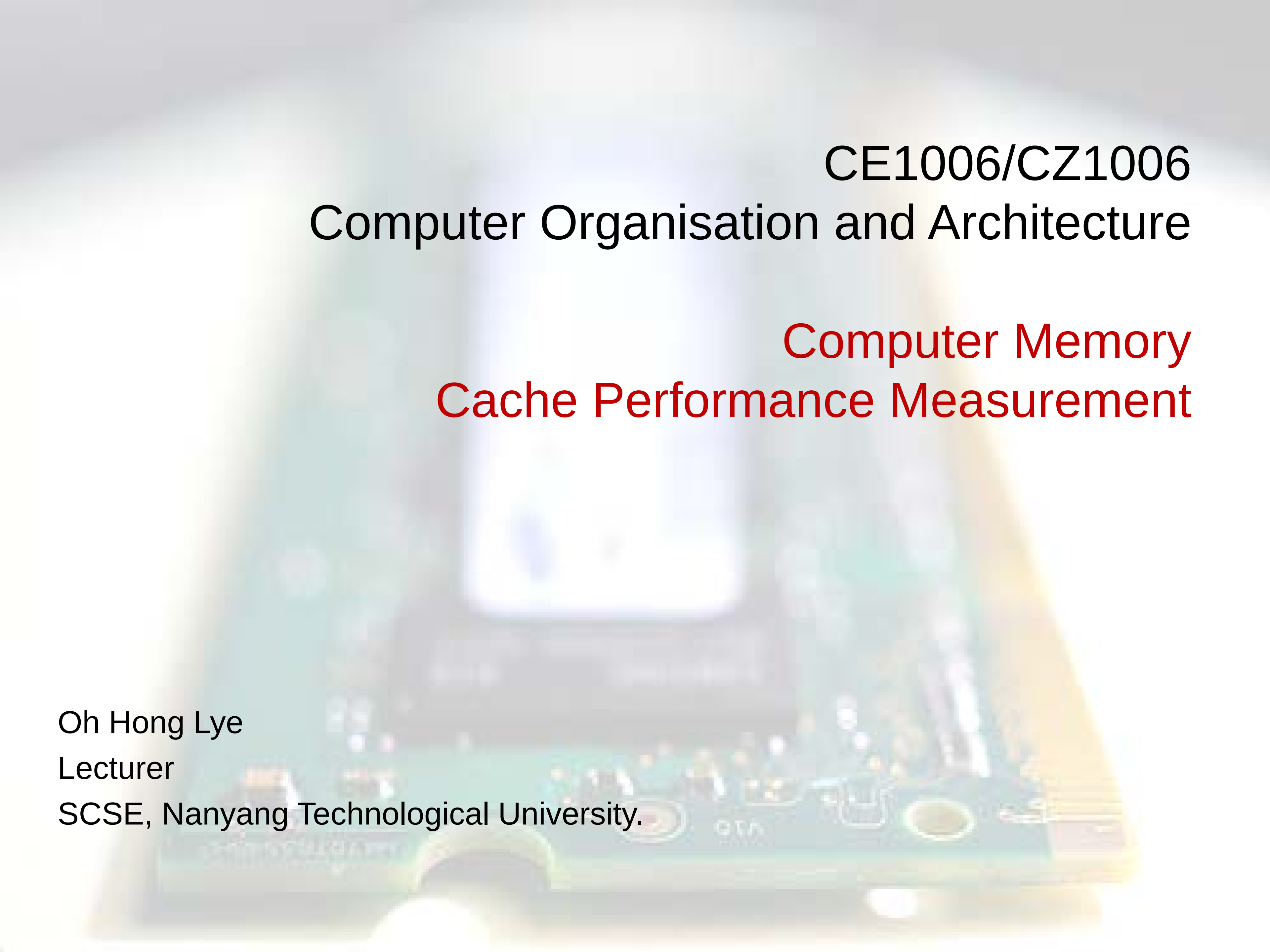
# of Blocks in Cache = 16/4 = 4 Blocks => 2 address bits (Block index)

# of TAG bits = 8-2-2 = 4 bits.

CE1006/CZ1006
Computer Organisation and Architecture

Computer Memory
Cache Performance Measurement

Oh Hong Lye
Lecturer
SCSE, Nanyang Technological University.



Cache Related Terminologies

- **Cache Hit**
  - Data is found in the cache.
- **Cache Miss**
  - Data is not found in the cache.
- **Cache Hit rate**
  - Percentage of time data found in the cache
- **Cache Miss rate**
  - Percentage of time data not found in cache.
- **Miss rate = 1 - Hit rate.**

Cache Related Terminologies

- **Cache hits**, **cache misses** and **cache hit rate** are common indicators of the effectiveness for the applied cache mapping scheme.

- The **hit time** is the time required to access data from cache.

- The **miss penalty** is the time required to process a miss, including
  - Time it takes to replace a block of memory, **PLUS**
  - Time it takes to deliver the data to the processor.

Effective Access Time

- The performance of hierarchical memory is measured by its **effective access time (EAT).**
- EAT is a weighted average that takes into account the hit ratio and relative access times of successive levels of memory.
- The EAT for a two-level memory is given by

  $\mathrm{EAT} = H \times \mathrm{Access}_C + (1-H) \times \textit{Miss Penalty}$

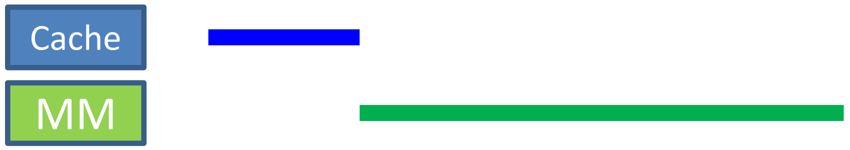
- For simplification sake, assume *Miss Penalty* = Access_C + Access_{MM}.

  $\mathrm{EAT} = H \times \mathrm{Access}_C + (1-H) \times (\mathrm{Access}_C + \mathrm{Access}_{MM})$

  where
  H is the cache hit rate
  Access_C access times for cache
  Access_{MM} access times for main memory

Effective Access Time (Example)

Sequential Access of Cache and Main Memory



- Consider a system with a **main memory access time of 200ns** supported by a **cache** having a **10ns access time** and a **hit rate of 99%.**
- If the **accesses do not overlap**,

  The EAT is:

  0.99(10ns) + 0.01(10ns + 200ns) = 9.9ns + 2.1ns = 12ns.
- This equation for determining the effective access time can be extended to any number of memory levels.