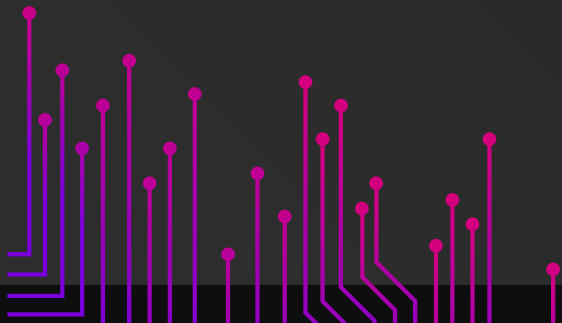
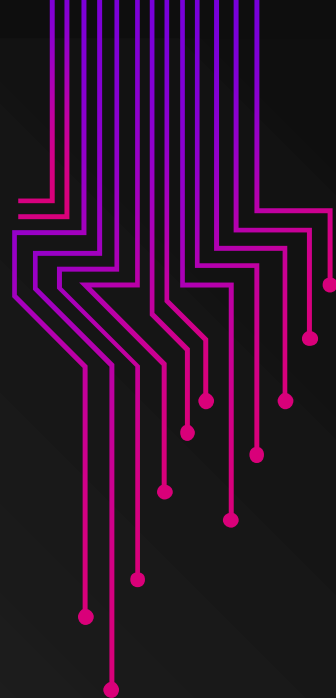


# Operating System 112 Fall

## Homework 4 - Cache Simulation

Prof. 蔡文錦

TA 林浩君, 薛乃仁, 周彥昀, 許承壹



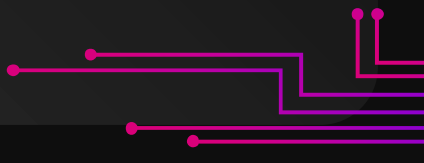
# Table of Contents

<b>1.</b>	Objective
<b>2.</b>	Part 1(30%)
<b>3.</b>	Part 2-1(35%)
<b>4.</b>	Part 2-2(35%)
<b>5.</b>	Submission and Rules
<b>6.</b>	Reference



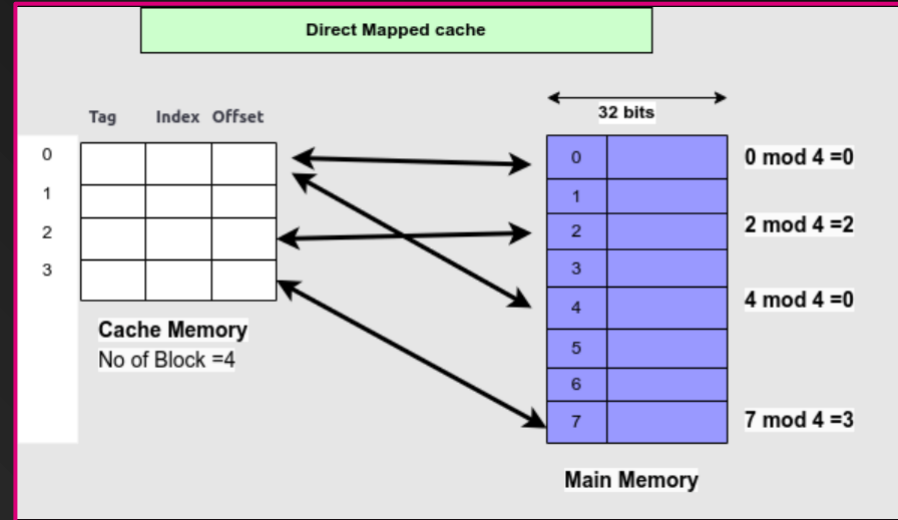
# Objective

In this homework, we are going to take a closer look at the functionality of cache, and try to do some simple simulation, with respect to some typical type of cache and block replacement policies.

- Block replacement policies are similar to what you have learnt of dealing with page faults.
  - There are physical addressed cache and also virtual addressed cache.
  - More details about cache are introduced in computer architecture.
- 

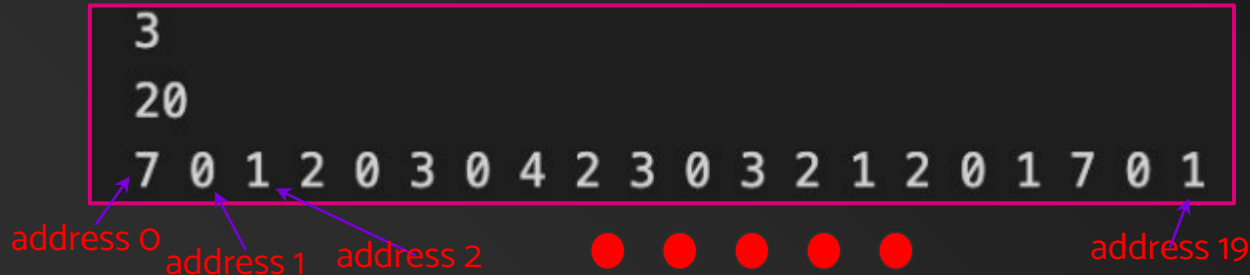
# Part 1: Direct Mapped Cache(30%)

- We are going to just focus on how data are stored in cache, without maintaining tag bit and offset.
- Objective: count the **total cache misses**.
  - You need to first traverse the whole cache to find the requested data
  - If not found, place it with respect to its address and count it as a cache miss.



# Part 1: Direct Mapped Cache(30%)

- Input:
  - Each test case contains three lines:
    - Number of cache lines/ blocks
    - Reference data stream size
    - Reference data (only integer numbers)
  - Use the index of input sequence as the address of the data.
- Please use **STDIN / STDOUT** to read data.
- Example:



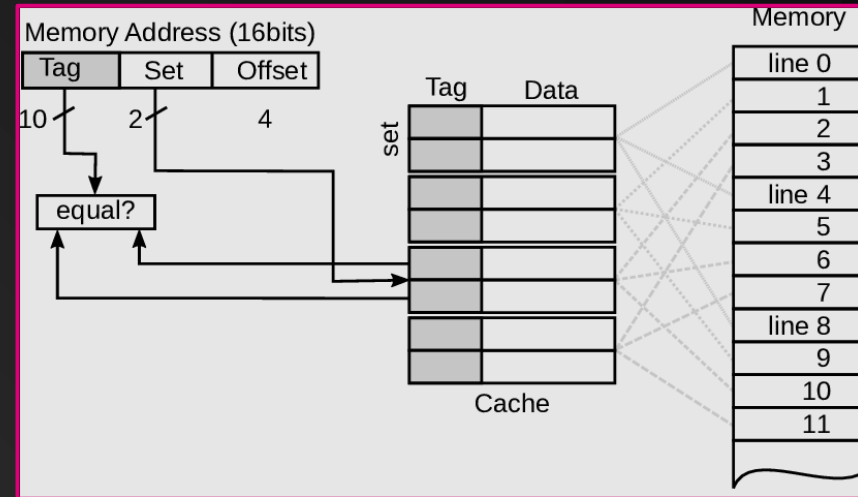
- Output: please follow the following format

**Total Cache Misses:10**

# Part 2: K-way Set Associative Cache

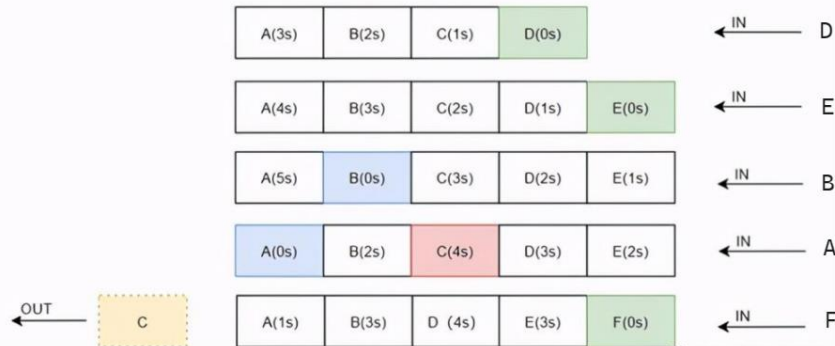
- We are going to design a **k-way set associative cache** with different block replacement policies, which are **LRU** and **LFU**. Where k is a number that determines the number of blocks of a set.
- Objective: count the **total cache misses**.
  - You need to first traverse the whole cache to find the requested data.
  - If not found, first determine which set it should be put in, and use the requested block replacement policy (LRU or LFU) to determine which block should be swapped out, and then write the new data in it. And count it as a cache miss.

Example of a 2-set associative cache



# Part 2-1: K-way Set Associative Cache with LRU(35%)

- Least Recently Used Algorithm(LRU):
  - If a set of blocks are full, then find the least recently accessed block and replace it with the new block.
  - If a set is not full, we just find the first empty place in the cache and write the data inside.
- Example: one of a set of 5-set associative cache with LRU



# Part 2-2: K-way Set Associative Cache with LFU(35%)

- Least Frequently Used Algorithm(LFU):
  - If a set of blocks are full, then find the least frequently accessed block and replace it with the new block.
    - You need to maintain the **frequency**(times it has been accessed) of the block.
    - If a block is swapped out, and enter the cache again, you need to calculate its frequency **from 0 again**.
    - If multiple blocks have the same frequency, then replace the one with the smallest index of the set.
  - If a set is not full, we just find the first empty place in the cache and write the data inside.
- Example: 5-set associative cache with LRU



# Part 2-2: K-way Set Associative Cache with LRU

- Example: 5-set associative cache with LRU



# Part 2: k-way Set Associative Cache

- Input:
  - Each test case contains four lines:
    - Number of cache lines/ blocks
    - K, which is the Number of cache lines/ blocks of a set
    - Reference data stream size
    - Reference data (only integer numbers)
  - Use the index of input sequence as the address of the data.
- Please use **STDIN / STDOUT** to read data.
- Example:

```
4
2
20
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
```

address 0   address 1   address 2   ●   ●   ●   ●   ●   address 19

- Output: please follow the following format

```
Total Cache Misses:11
```

# Submission and Rules

## Rules:

- Please use **C++** to implement.
- No **plagiarism** is allowed, since the grade of this course is critical for **graduate program application in CS related field**, we will not pardon such behavior at all, so please be responsible to yourself. You can discuss with your classmates, but don't copy and paste.
- Incorrect filename / file format will get -10% point.
- Delayed submission will get -20% point per day.

## Submission:

- Please follow the naming rules:
  - **4-1.cpp** for part 1.
  - **4-2\_LRU.cpp** for part 2-1 with LRU algorithm.
  - **4-2\_LFU.cpp** for part 2-2 with LFU algorithm.
- Please compress the above three files into one .zip file with the format of **{studentID}\_HW4.zip** (e.g. **312551062\_HW4.zip**)
- There will be five test cases of each part, two are given, three are private.