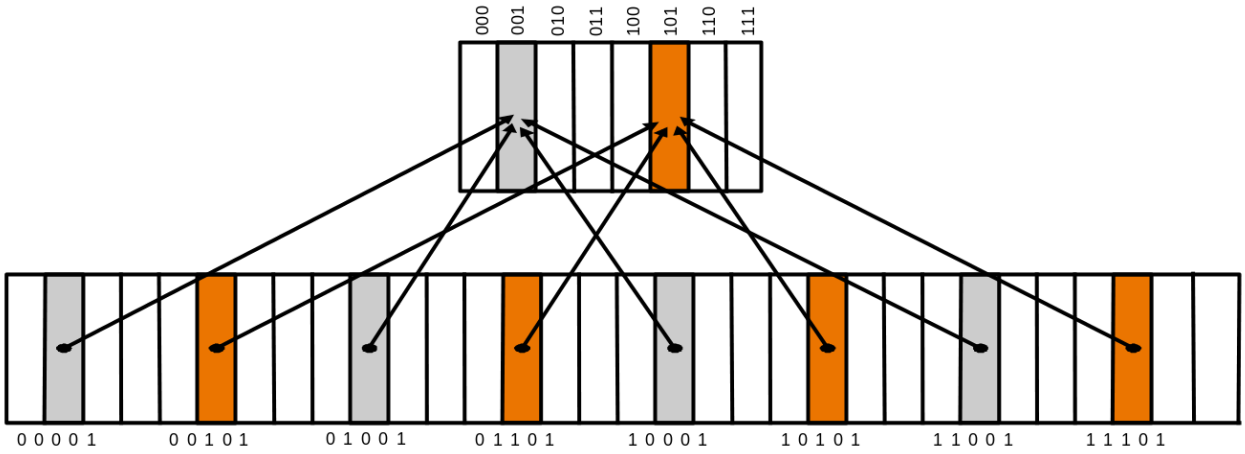
0511105 李頤

CO Lab 6
Cache Simulation

I. Cache Basics

此次LAB是用來模擬各種參數對cache mapping的效能的影響。sw, lw等是負責從memory存取資料的指令，然而存取memory非常花時間，所以將一些常用的資料存在cache是提升效能的好發法。cache的空間有限，所以勢必要將單一位址map到許多memory的位址，最基本的方式是direct mapping



為了提升效率有了n-way set associativity，分割的方式如下

One-way set associative
(direct mapped)

Block	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

Two-way set associative

Set	Tag	Data	Tag	Data
0				
1				
2				
3				

Four-way set associative

Set	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0								
1								

Eight-way set associative (fully associative)

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data

II. Code Explained

我用原本的函數去改，新函數叫做 `simulate_ass` （好名字），多了一個參數用來設定associativity

offset bit是用來指block內的位址
index bit是用來指cache的address
line表示這個cache有幾個set

因為要實現associativity，所以我將原本的cache_content改成二維陣列
cache[i][j], i是指line, j是指line裡面哪個分支（？）
尋找方式先一index找到哪一個line(or set)，然後再去比對每一個tag決定miss or hit。

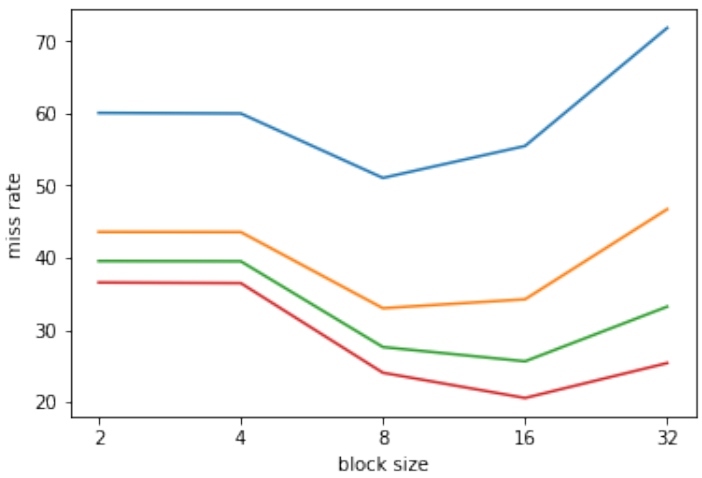
LRU的實現我花了蠻多時間的，我用deque宣告一個LRU二維陣列，用來記錄錄最近被用到的block順序。
以一個4-way associativity為例，一個line面會有4個block(tag+data)
當一個block(tag+data)被叫到時，這個block的index會從deque裡被抽出然被丟到deque的最右邊，變成最近用到的。
當一個line已經滿了，就從deque最左邊抓出最沒被用到的block index，然後把它指到的block置換掉。因為此block index是最近被access到的，所以會被丟到最右邊，最左會被pop掉

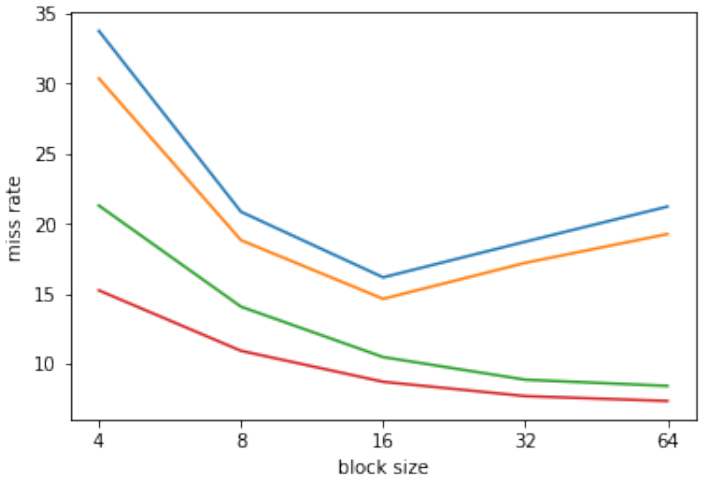
Least Used	…	…	Most Used

最後的圖我是複製C++跑出來的結果丟到matplotlib畫的（比較方便），結果趨勢雖然正確（好像？）但是數值有點怪怪的，希望是對的

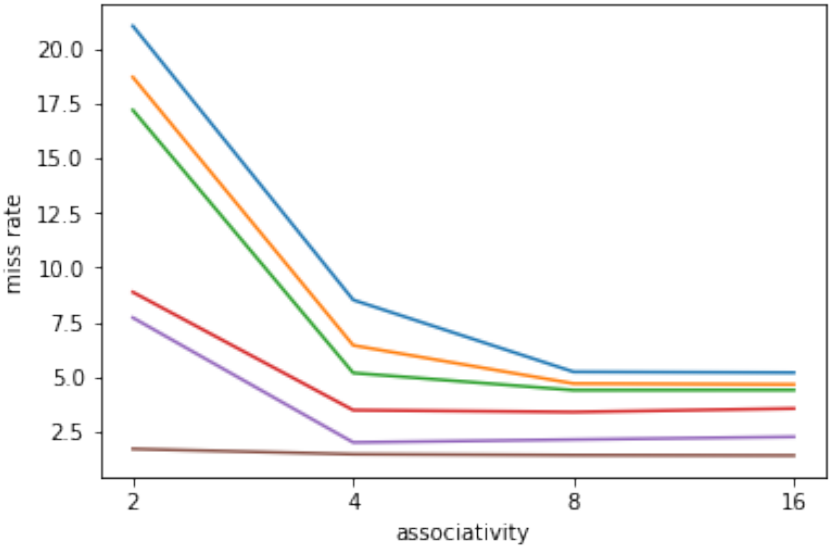
III. Result & Observation

a.Fixed associativity





左圖由上而下分別是cache size = 64, 128, 256, 516 Byte
因為miss rate怪怪的，所以我改跑以下數據
右圖由上而下分別是cache size = 2K, 4K, 16K, 32K Byte，感覺比較正常（？）
圖形的趨勢會呈現上圖的情況是因為block size增大雖然會load進比較多資料，理論上應該可以減少miss rate，但是block變大也表示line變少了，分散的資料不容易在同一個block找到，所以又增加了miss rate



由上到下是 cache size = 1K, 2K, 4K, 8K, 16K, 32K
雖然line的數量也會因為associativity增加而減少，但是因為是不同tag，所以反而涵蓋比較大範圍的資料，miss rate也因此降低，但實際上會因為comparator而減慢速度所以也不是fully-associtativity就一定比較好。

0511105 李頤

Reference & 心得

強大的網站讓我了解cache在尬麻
https://www.ntu.edu.sg/home/smitha/ParaCache/Paracache/sa2.html
這次的lab讓我終於搞懂(應該啦)cache mapping和set-associativity 了