**專題一報告**

課程主題： 計算機結構

學號：A1095551

姓名：廖怡誠

本次專題的內容為設計Cache Controller，模擬在不同Cache Size、Block Size與n的情況下，Cache內部會如何操作，以處理Processor所給予的記憶位置。

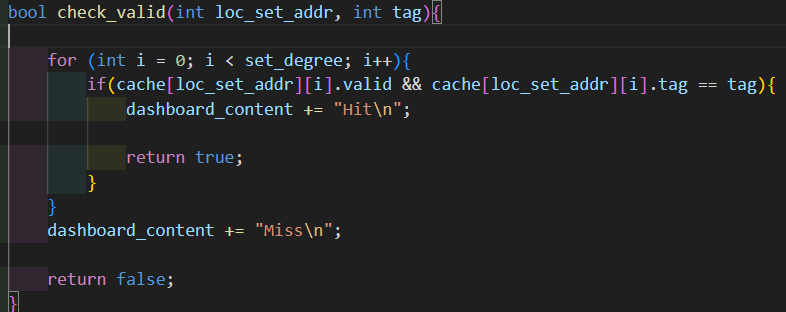
1. **架構設計**

在Cache的設計中，我在Class cache\_controller{}建立時會根據Cache Size、Block Size與n建立一個二維陣列Cache，column為set的數量、row為大小為n的一維陣列作為cache block，每個cache block皆會有valid、tag、data、time四個變數，其中time為記錄LRU所設計的計數器。



圖一、cache建構

在收到處理器提供的記憶體位置，並計算對應的block number後，使用for-loop檢查對應block number的陣列中是否有對應的tag能夠Hit，反之則為Miss。



圖二、確認set中所有valid的函式

1. **遭遇問題**
   1. **Cache block設計**

原先Cache block的資料結構想要使用priority queue來實作，因為priority queue能夠以Time調整優先度，但如果使用priority queue進行節點尋訪以更新Time次數與檢查valid，與使用同為不連續記憶體的vector相比較為不便，因此最後決定以vector為Cache block的資料結構，並加上能夠判斷LRU的機制作為輔助。

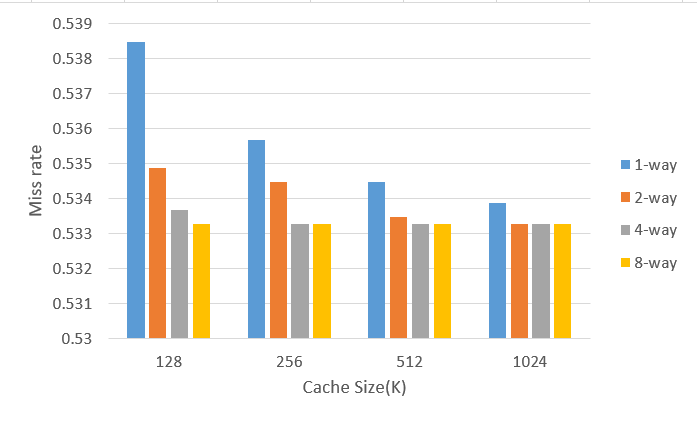
* 1. **測資測試問題**

本次只需要輸出miss rate，其他的LRU替換與tag數值都不需確認，而我注意到假設我寫錯計算tag或LRU替換block的方法，對於miss rate的數值是不會有所改變。首先，針對tag計算錯誤問題，由於我是先計算要填入哪個set，因此即使tag錯誤，填入的位置也仍會相同，差別只在於紀錄的tag與正解不同，並不會影響miss rate。而LRU替換問題，我最初的錯誤作法是將目前cache block第一個block取代，此方式與LRU有些微的區別，因為如果有hit的話，第一個位置的data不一定會被取代，而碰巧測試資料的address都為連續出現，這會使無論取代哪個data，Hit的次數都不會被影響。

1. **實驗結果分析**

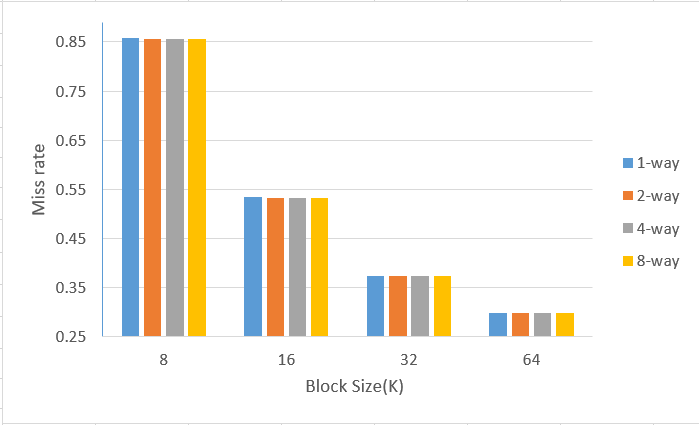
**Case 1: 固定Block Size為16B，不同Cache size對Miss Rate的影響。**

Cache Size越大Miss Rate越小，way越大Miss Rate越小，但n越大，Cache Size對Miss Rate的影響也越小。

****

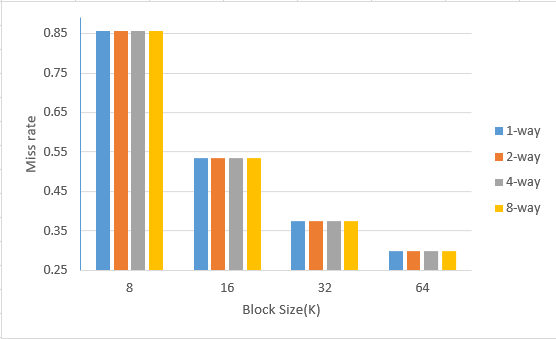
**Case 2: 固定Cache Size為512K，不同Block size對Miss Rate的影響。**

Block Size越大Miss Rate越小，然後way越大對於Miss Rate的影響不大。

****

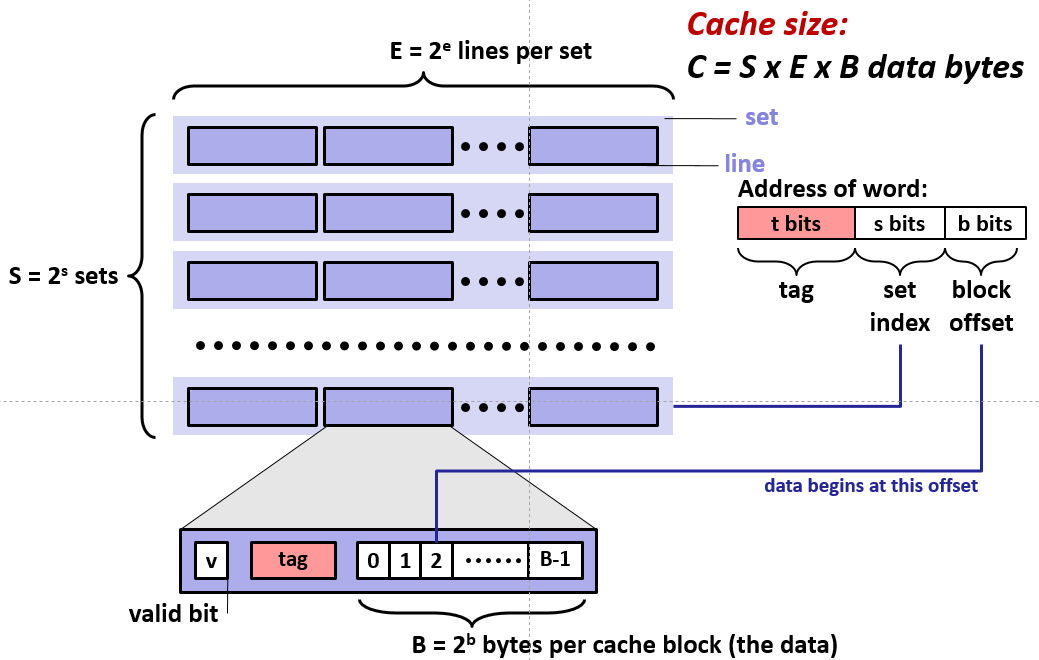
**Case 3: 固定Cache Size為1024K，不同Block size對Miss Rate的影響。**

與Cache Size為512K的結果差異不大，Block Size越大Miss Rate越小，然後way越大對於Miss Rate的影響不大。

****

1. **心得**

在這次的實作中，我充分的了解cache在存取data是如何運作，整體實作的時間並不會花費太長，主要的時間都在理解cache內部的架構該如何設計，像是上課投影片的圖片中，一個set只有一個valid，但如果只有一個valid的話，就沒辦法判斷N-way的情況，因此我花費蠻多時間確認實際的架構為何，最後決定以圖三中架構實作。



圖三、cache架構

而這次專題，我嘗試使用C++來實作，因為我認為此類的底層架構使用Python實作不太合適，C語言在Linux kernel中被大量運用，且編譯、執行的速度也較Python優異，雖然在實作上相較於Python複雜，但也使我對於這類的實作更加的熟悉。