**Computer Organization 2018**

**HOMEWORK 4**

系級: 108資訊 學號: C14041141 姓名: 王韋凱

**問題(Question)**

Q1. How do you know the number of block from input file?

number of block = cache size / block size

由於 cache size 的單位 是KB ， block size 的單位是B

因此要先將cache size 乘以 K(1024)才能除以block size

得到正確的 number of block

Q2. How do you know how many set in this cache?

number of set =number of block / number of way

directed-map相當於one-way set associativity，因此number of set =number of block / 1 也就是 number of block

在four-way associativity的case中，number of way = 4因此number of set =number of block / 4

在 full-way associativity的case中，就是number of way=number of block即number of set = number of block / number of block=1

Q3. How do you know the bits of the width of the Tag ?

由於 width of memory 的size 為32bit，而32 = width of offset + width of index + width of tag ，因此width of tag =32 - width of offset - width of index

Q4. Briefly describe your data structure of your cache.

int valid => 將有trace進入過的cache 設為1 反之為0

unsigned long tag => 判斷where a block stored in a cache

int used\_turn => 讓fifo 及lru 判斷

Q5. Briefly describe your algorithm of the implementation of LRU.

每個 trace of memory access 時，將代表他是第幾個進到cache的變數(turn)，記錄到該cache中的used\_turn，因此used\_turn越大，代表該trace越晚進入cache，而下一個trace在進入cache時只要找出最小used\_turn (最早進入)的cache，就可以進入該cache了

**心得(Report)**

(請寫下完成本次作業的心得、學到哪些東西、困難點的部分。)

(Please write your learned lesson and conclusion, and difficult point.)

在實作這次的作業過程中，讓我更了解cache的架構，包括valid以及tag等等，而最難的部分是要如何將看似一樣的演算法FIFO以及LRU實做出來，但也能夠複習一下計算機概論學的東西XD，而將trace2.txt和trace3.txt當作input時，會發現trace3跑的比trace2慢，我想是因為trace3是full-way associativity 而 trace2是four-way associativity，因此trace3在每個trace access 時要檢查每個block 而 trace2 只要檢查 1/4 倍的block就可以了