**Computer Organization 2019**

**HOMEWORK 6**

系級: 資訊110 學號: F74065042 姓名: 李浩榮

**問題(Question)**

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

解答:

對於DirectedMapped,4\_Way Associativity:

* DiredtedMapped 直接只有一個block(一個column)
* 4\_Way 直接只有四個blocks(四個column)

對於Fully Associativity:

* 因為只有一個set(一個row),所以

blockNumber = Cache\_Size/Block\_Size

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

對於DirectedMapped Associativity:

* Set = Cache\_Size / (Block\_Size\*1)

對於4\_way Associativity:

* Set = Cache\_Size / (Block\_Size\*4)

對於Fully Associativity:

* Fully 就直接一個set

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

對於DirectedMapped Associativity:

* 2^offset = Block\_Size ,已知Block\_Size,即可找出offset
* 2^indexBits = index,承上題,已知set數=index數,即可找出indexBits
* tagBits = 32 – offset – indexBits

對於4\_way Associativity:

* 找法也和DirectedMapped一樣,但是找index的時候記得要除四,如上題

對於Fully Associativity:

* Fully的set只有一個所以,2^0 = 1,所以Fully的indexBits 直接等於1
* 2^offset = Block\_Size ,已知Block\_Size,即可找出offset
* tagBits = 32 – offset - indexBits

Q4. Briefly describe your data structure of your cache.

* 我的cache是用vector來構成的二維陣列,它其實就是一個linked-list,可以動態增刪.
* 對於directedMapped 和 Fully 我是只給它一維就夠了,對於4\_way是用二維來執行的
* 通過push\_back(),erase(),begin()等vector library中的內建函式來完成以上的工作

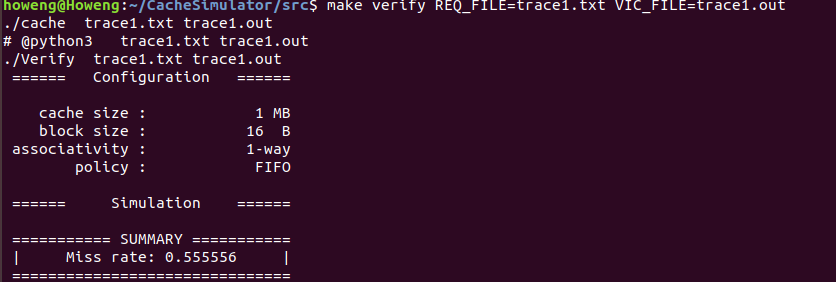
Q5. Briefly describe your algorithm of LRU.

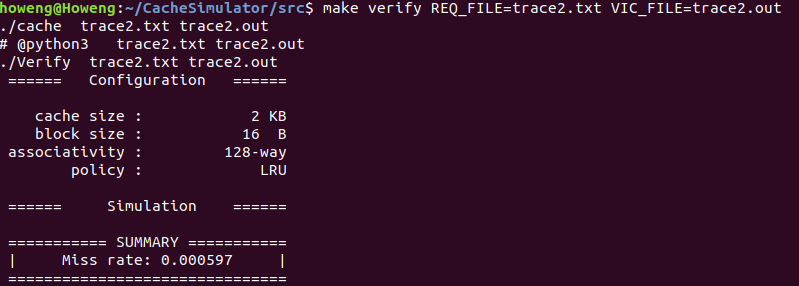
* 我是用vector把資料讀進來並串成一條鏈.
* 第一進來的就在cache[0],第二個就是cache[1],以此類推
* 所以越前面的就是進來越久的,越後面的就是越剛進來的.
* 所以如果要踢掉的話,就先踢掉第一個,並且把新的加入到鏈的尾端
* 整個LRU是先檢查這條鏈中有沒有一樣的tagValue(Hit),有的話就覆蓋且把它移到最尾端(刷新已進入時間的概念),有執行以上步驟的話就用goto跳過以下的步驟直接去做下一筆測資.
* 否則再去判斷裡面有沒有空位,有的話就放下去
* 如果沒有Hit,也沒有空位,那就要踢人了,所以就把最前面的也就是cache[0].begin() 這個erase掉,並把新的tagValue加入到尾端.

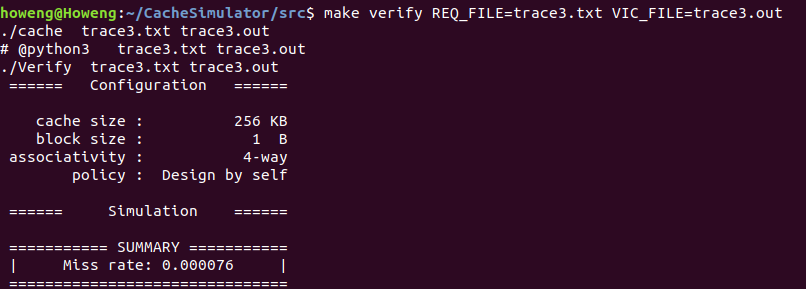
Q6. Briefly describe your algorithm of your policy.

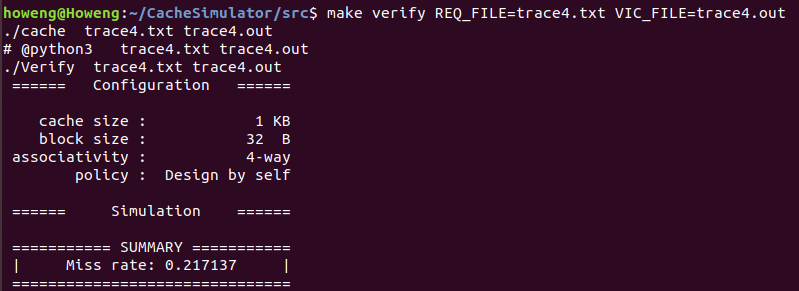
* 其實就是LRU

Q7. Run trace2.txt, trace3.txt and then makefile to get the miss rate and put it in your report.









**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.)

* 最後一個作業了,其實如我之前所說的,根本沒有難,但是很煩==,總的來說很有趣,蠻喜歡的
* 學到了cache的結構以及如何變形(directedMapped,4\_way,Fully)
* 接下來就說說看遇到的問題以及解決方法:
  + 最容易犯的錯誤就是offsetBits,tagBits,indexBits弄錯,如何去找出這些值上面已經有提過了就不多解釋.這部分很重要,錯了的話後面就不用做了.比較特別的就是fully的indexBits要直接設為0(因為fully只有一個set,2^indexBits = set)
  + 活用vector會很方便,我原本是另外用一個陣列來存資料進來的時間點,中間很複雜,有很多步驟要做,也有很多變數.但是最後我換成了用vector先進先出的靈活特性來完成這個FIFO和LRU的部分.
  + 學到了FIFO和LRU最大的差別就是Hit的時候,LRU要刷新時間,也就是把第一個拿掉(舊),放到最後面(新).但是FIFO不用,就照常跑就好,只當要替換元素的時候,才需要兩個都pop掉第一個,放新的tagValue進來鏈的尾端.所以LRU和FIFO的寫法也可以寫在一起.
  + 多活用並建立一個共同function,來讓你的code更整齊,更方便.
  + 遇到的兩個bug:
    - .out檔案的最後一筆不能換行!!!(害我debug花了很多時間)
    - 宣告出來的vector一定要用,不然也會不過…原本只是宣告了,但是後來沒用到就一直run不過了,所以要注意.
    - 總結來說這個作業如果遇到segmentation fault不用想一定是空間出了問題,例如access到了你沒和記憶體要的地方.特別是使用vector的時候,如果如果push\_back()兩次而已,但是for迴圈會access到大於2次的地方就segmentation fault了.要小心!