Lab5_Report

0710028 陳敬諺

1. Code and Calculation Explanation

Total memory stall cycle:

Fig 1.(a): Hit 時，address 傳到 cache，address 傳入 cache 內，再回傳，所經過的 memory stall cycle (delay)為 1+2+1=4。而 miss 時，memory 傳到 cache 要經過一次傳送 address，一次傳送 data，一次 access memory content，一次 access single cache content，共 1+100+1+2=104 個 delay，但因為 cache 抓 8 words 會依序傳送所以一次 miss 在 memory 與 cache 間要等 8*104 個 delay，再加上一次 processor 與 cache 間的 4 個 delay，所以列出算式為
(1+2+1)*(access-miss)+miss*(1+8*(1+100+1+2)+2+1)

Fig 1.(b): 和(a)的算法類似，但由於 memory 到 cache 一次能傳送 8 words 所以一次 miss 的 delay 縮減為 4+104，算式列為
(1+2+1)*(access-miss)+miss*(1+(1+100+2+1)+2+1)

Fig 1.(c): 若在 L1 中 hit，所經 delay 改為 1+1+1=3，而 L1 中 miss 在 L2 時 hit，L1 與 L2 間的 delay 為 1+10+1+1，但要依序傳送 4 words，所以乘以 4，之後再加上 L1 與 processor 間的 delay 有 3，要是 L2 中也為 miss，則為講義上的算法，delay 為 1+32*(1+100+1+10)+4*(1+10+1+1)+1+1
所以算式記為
miss_2 * (1+32*(1+100+1+10)+4*(1+10+1+1)+1+1)+
(access_1_2 - miss_2) * (1+4*(1+10+1+1)+1+1)+
(access - access_1_2) * (1+1+1)

Execution:

從 assembly code 中可以看出在 k 層的 loop 之中包含了 22 行指令，j 層則除了原來 k 層的指令再加上 5 行，且 k 層迴圈結束時，會先跑過 k 層的開頭兩行，j 層所經過的指令數會是 5+k*22+2，同理 i 層要經過的指令數 5+j*(5+k*22+2)+2，最後是外部開頭的兩行指令與 i 層終止時的兩行指令與最後的運算終止，所以會是 2+i*(5+j*(5+k*22+2)+2)+2+1。

Code:

Fig 1.(a)(b)兩者的 code 在模擬 cache 的行為上是相同的，只是最後結算 total memory stall cycle 所用的算式不同，因為 memory 與 cache 間傳輸的負載不同。

Fig 1.(c)在 L1 cache 中出現 miss 之後必須先 access L2 cache，若又是 miss，才到 memory，在 code 的實現上，是在兩個 cache 的架構是一樣的，即讓在 L1 中 miss 發生的 address 立即丟入 L2 運作。

2. Discussion

   以下是 4 筆測資所算出來的 total memory stall cycles，順序依序為
   fig 1.(a) fig 1.(b) fig 1.(c):

   c1:

   6016 1648 12144

   c2:

   19072 5968 32736

   c3:

   7626560 1068008 791552

   c4:

   251426432 35098320 998152040

   由上面的資料可大概看出，在上面測資測試下，(b)相對(a)都是表現較好的，因為一次傳送多筆資料使得 miss penalty 在相同的情況下被減少許多，另外，當在資料量少的時候，(c)的效率明顯較前兩者差許多，因為(c)在讀取 memory data 會花大量時間取得多筆資料，但在 c3 的測資中可明顯發現(c)優於另外兩者，可在 c4 的效率卻又小於另外兩者，推測(c)的架構，在資料大量且集中的時候使用效益會是最好的。

3. Experience

   延續 lab4，lab5 在實現上並不困難，多半是了解 fig 1.(c)的架構後，某幾段的 code 複製貼上，修改參數就好，但我一直到做完才發現一個很嚴重的問題，我 lab4 的 n-ways associative 的架構是錯的，我 tag 的 bit 忘記隨 set 做更動，因為當時做出來的結果合幾個朋友的結果是相同的，所以就沒多想，現在才知道自己錯的多離譜，看來上次對於 cache 的架構那部分並未熟讀過。