計算機結構程式專題(一) 報告

A1055547 王公志

一、使用程式語言、環境

甲、程式語言：Python

乙、開發環境：Google Colab

丙、測試資料：trace.txt (5003 筆測資)

二、LRU 實作方式：

甲、將 cache 中每個 block 賦予一個 counter(初始值為 0)，counter 的值越大，表示越久沒有使用；反之，值越小表示該 block 離最近一次使用越近。

乙、每當有新資料讀入時，將 cache 中所有 invalid block 的 counter 加一。

丙、若是 cache hit 則將該 block 的 counter 設為 1，表示該 block 為最新使用的。

丁、當 set full，且 tag 與 set 中的所有 block tag 相異時，取代 set 中 counter 最大的 block，並將該 block 設為 1。

三、心得與討論：

甲、每當 associativity 提高時，miss rate 理應隨之下降，但是觀察模擬結果，當 associativity 由 4 提高到 8 時，miss rate 並不會下降，推測是因為測資數量太少，且散落於各個 set 中，造成 conflict misses 沒有隨之減少。

乙、當 Cache Size 和 associativity 固定，並逐步提高 Block Size 後，可以發現每次 miss rate 伴隨顯著的下降。該情形說明測資的 address 其位置相近，因此當提高 Block Size 後，由於 Spatial Locality 造成 Compulsory missess 的減少。

丙、當 Block Size 和 associativity 固定，並逐步提高 Cache Size 後，miss rate 雖有下降，但是幅度不大，miss rate 仍在 0.53 左右。該情形可以解釋為 Capacity miss 沒有太大影響，可能的原因在於測資數量不多，且 address 散布在各個 set 中，且各個 address 之間的距離不會太遠，使得 set 不會被佔滿；又由於當 Cache Size 在較小時就沒被佔滿，因此當 Cache Size 變大後只有 set 數跟著變多，故各個 set 的 block 只會變少，使得 Capacity miss 影響更小。

四、模擬結果：

Cache Size (Byte)	Block Size (Byte)	n-way	Miss Rate(*紅色表示下降、藍色表示不變*)
128K	16	1	0.538477
		2	0.534879
		4	0.533680
		8	0.533280
256K	16	1	0.535679
		2	0.534479
		4	0.533280
		8	0.533280
512K	16	1	0.534479
		2	0.533480
		4	0.533280
		8	0.533280
1024K	16	1	0.533880
		2	0.533280
		4	0.533280
		8	0.533280
512K	8	1	0.857885
		2	0.856286
		4	0.856086
		8	0.856086
512K	16	1	0.534479
		2	0.533480
		4	0.533280
		8	0.533280
512K	32	1	0.374775
		2	0.373976
		4	0.373776
		8	0.373776
512K	64	1	0.299220
		2	0.297821
		4	0.297621
		8	0.297621
1024K	8	1	0.857086
		2	0.856086

		4	0.856086
		8	0.856086
1024K	16	1	0.533880
		2	0.533280
		4	0.533280
		8	0.533280
1024K	32	1	0.374175
		2	0.373776
		4	0.373776
		8	0.373776
1024K	64	1	0.298021
		2	0.297621
		4	0.297621
		8	0.297621

固定 Block Size、Set Degress

128K	32	1	0.380372
		2	0.375775
		4	0.375375
		8	0.373976
256K	32	1	0.376974
		2	0.375175
		4	0.373776
		8	0.373776
512K	32	1	0.374775
		2	0.373976
		4	0.373776
		8	0.373776
1024K	32	1	0.374175
		2	0.373776
		4	0.373776
		8	0.373776

128K	64	1	0.306616
		2	0.302419
		4	0.301819
		8	0.299420

256K	64	1	0.302219
		2	0.299620
		4	0.297621
		8	0.297621
512K	64	1	0.299220
		2	0.297821
		4	0.297621
		8	0.297621
1024K	64	1	0.298021
		2	0.297621
		4	0.297621
		8	0.297621