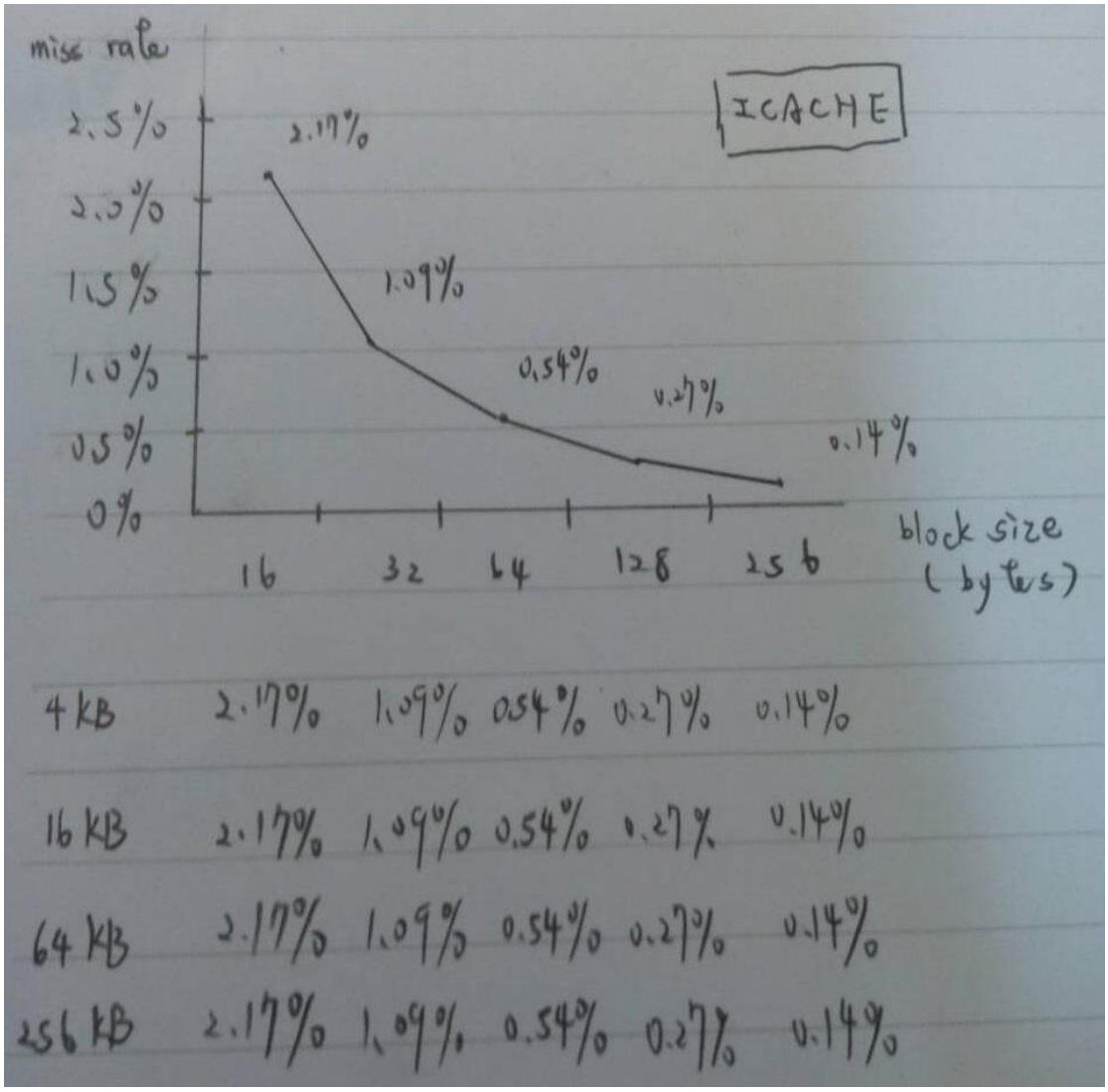
Computer Organization

1. Basic result:

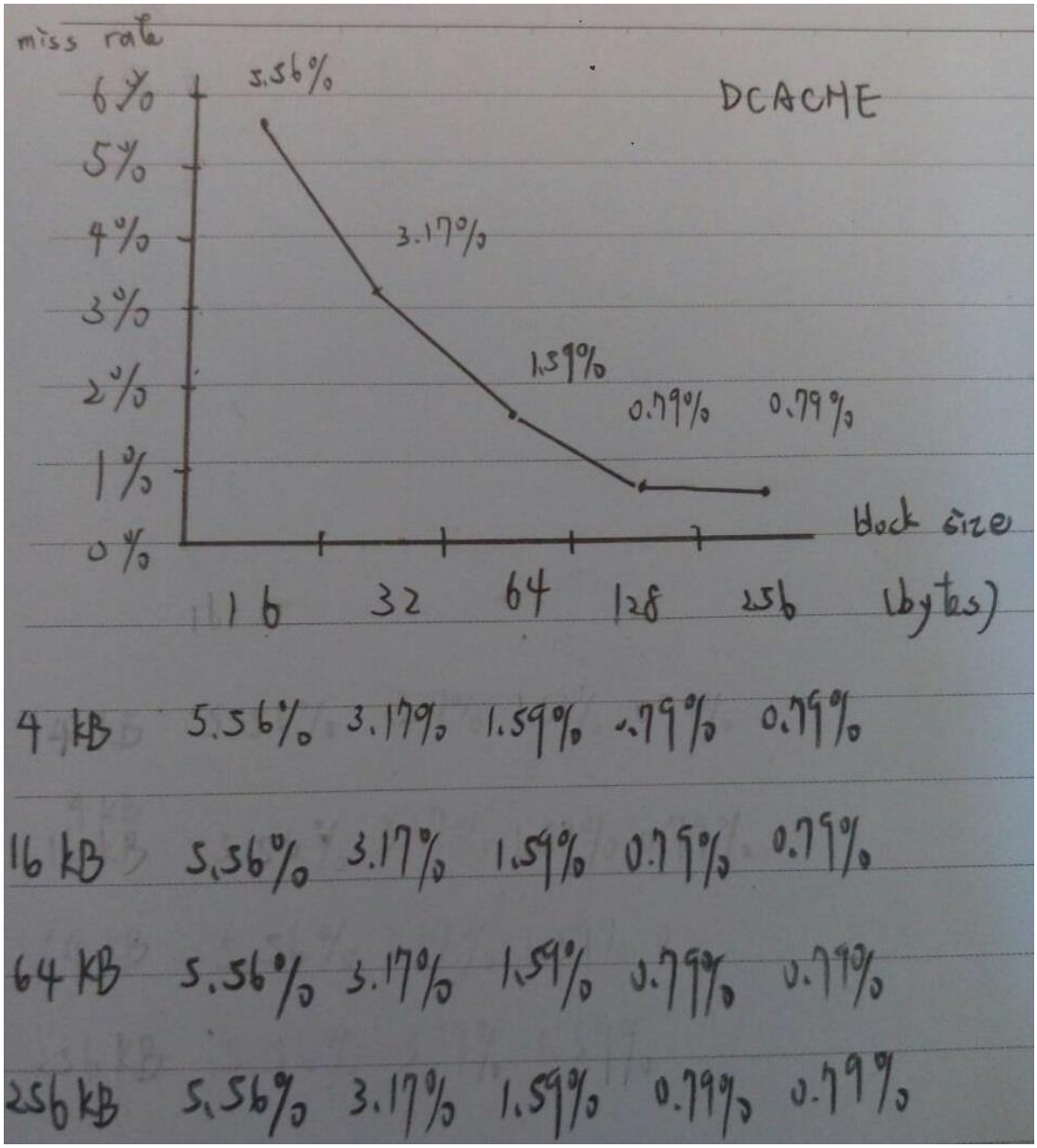
ICACHE:



由圖可知，在 **cache size** 一樣時，**block size** 增加則 **miss rate** 下降，因為 **block size** 增加意味著一個 **block** 能存更多的東西而較不容易發生 **miss**。

Computer Organization

DCACHE:

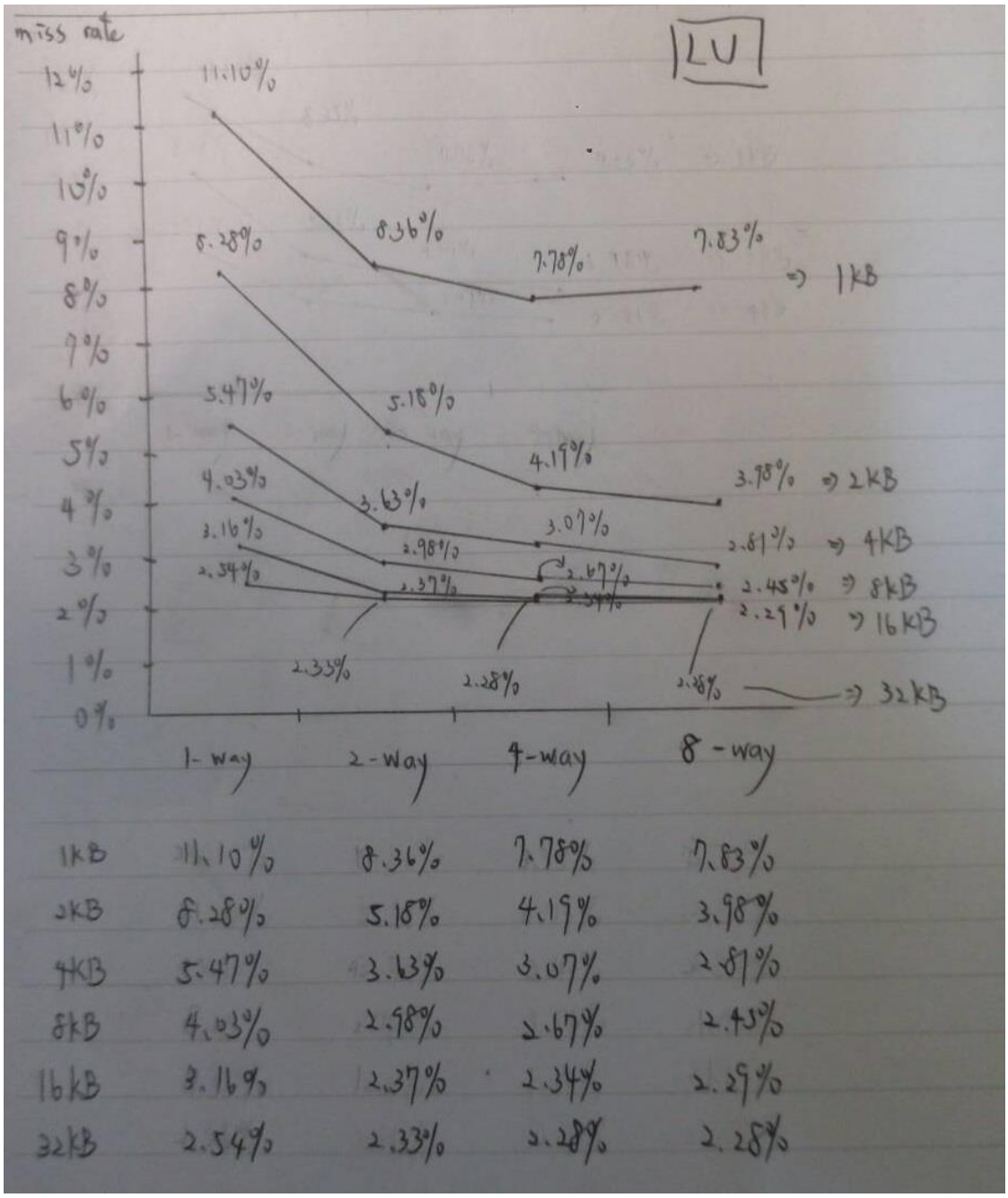


由圖可知，如同 **ICACHE**，在 **cache size** 不變時，當 **block size** 增加，**miss rate** 降低。其原理與 **ICACHE** 相同。

Computer Organization

2. Advanced result:

LU:



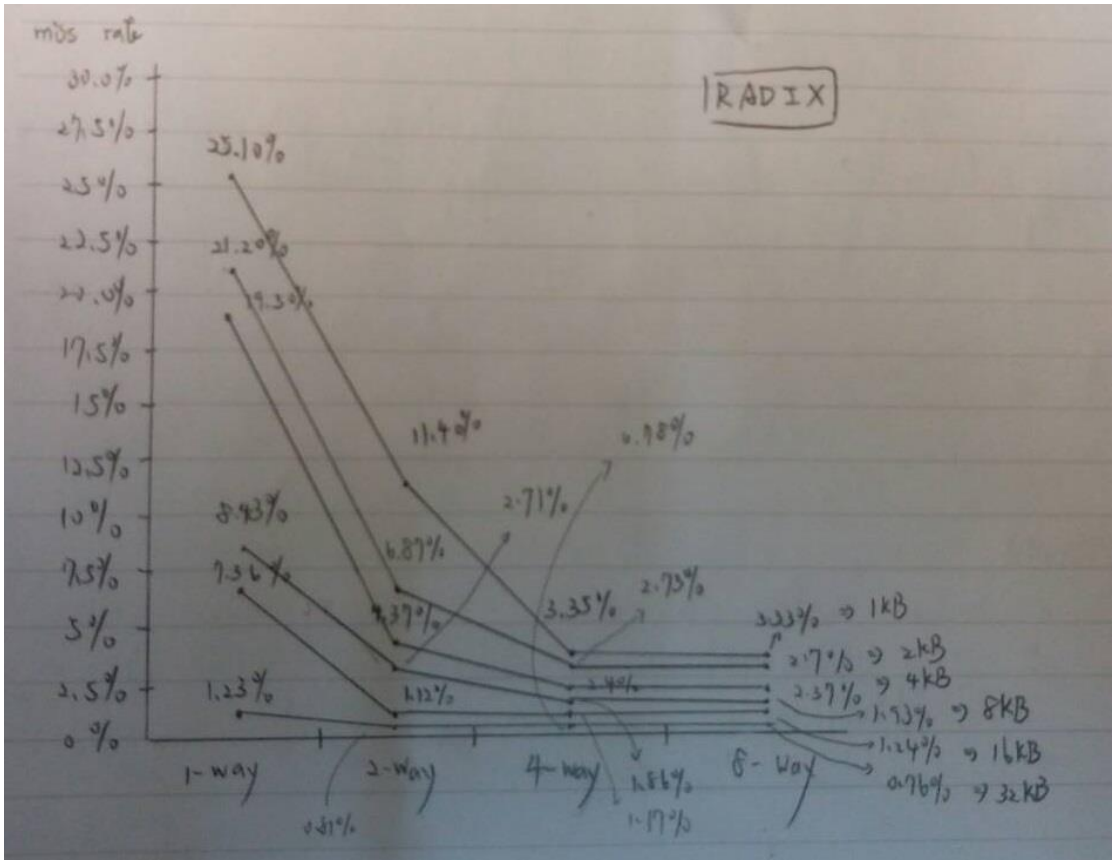
1kB	11.10%	8.36%	7.78%	7.83%
2kB	8.28%	5.18%	4.19%	3.98%
4KB	5.47%	3.63%	3.07%	2.81%
8kB	4.03%	2.98%	2.67%	2.45%
16kB	3.16%	2.37%	2.34%	2.29%
32kB	2.54%	2.33%	2.28%	2.28%

由圖可知，在 **cache size** 一樣的情況下，當 **associativity** 增加，**miss rate** 會降低。**Associativity** 的增加表示單位 **set** 能夠使用的 **block** 數變多，故 **miss rate** 下降。而當相同 **associativity** 時，當 **cache size** 增

Computer Organization

加，**miss rate** 下降，因為當 **cache** 能存的東西變多時，**capacity miss** 的機率下降，使得 **miss rate** 也下降。

RADIX:



	1-way	2-way	4-way	8-way
1kB	25.10%	11.40%	3.35%	3.33%
2kB	21.20%	6.87%	2.73%	2.70%
4kB	19.30%	4.37%	2.40%	2.37%
8kB	8.43%	2.71%	1.86%	1.93%
16kB	7.36%	1.12%	1.17%	1.24%
32kB	1.23%	0.81%	0.78%	0.76%

RADIX 的概念與 LU 相同，皆在 cache size 或 associativity 上升時 miss rate 會下降。

3. Detailed description of the implementation:

這次 LAB，basic 的部分基本上是沿用 LAB3 的 CPU 架構來時做出來的，direct_mapped_cache.cpp 的部分最主要就是改 cache size 及 block size 和 input 檔案的名字來做出來的，Advanced 的部分，我是用 direct_mapped_cache.cpp 的原型來打出來的，主要改 simulate 的 function 及新增一個變數 r(reference) 在 cache struct 這兩部分，simulate function 我加一個參數做代表 n-way associative 的 n，而若 hit，則除了 hit 的那部分的 r(記錄時間)要減少。而當 full 時，則從 r 中找出最小的值(LRU)，並取代他。最後的 miss rate 則為 miss 的次數除以總次數。

4. Problems encountered and solutions:

這次碰到最主要的問題是剛看到作業時有點傻眼不知該如何起手，再加上考完期末之後感覺有點智商砍半，看不太懂題目，所以多繞了一些路才找到 LAB5 要大家作的方向。

5. Lesson learnt (if any):

經過這次的 LAB 之後，我們覺得對 cache 有更深一層的領悟，除了

Computer Organization

direct_mapped one way 本身 miss rate 隨 cache size 和 block size 變化，更親自見證了 n-way 下的情況，感覺真是受益良多。