CA HW4 Report

備註：有在L1cache.h新增一個counter變數，用來在每做一次access的時候加一放回release bit，在做ＬＲＵ的時候比較release bit的大小來決定要拿走哪一行.

Q1.Flow chart

If it is not full in the block,just get data from the memory and return the data in cache.

If it is full here.First we check the dirty bit. (the data has been modified or not)  
If it has been modified ,do write back.(Use LRU).At the end , return the data in cache.

It will be miss here.We need to dertermine the block is full or not.

To check valid bit and tag bit.If

both of them are equal which means hit and return the data in cache

Get the address and do some arithmetic to get the offset, set\_total,index,block\_off set and address tag.

Q2-1

1. Quicksort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Direct map  way\_number=1 | 2-way way\_number=2 | 4-way way\_number=4 | Full associative way\_number=8 |
| Hit rate | 88.99% | 91.19% | 90.92% | 90.73% |

1. Mergesort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Direct map  way\_number=1 | 2-way way\_number=2 | 4-way way\_number=4 | Full associative way\_number=8 |
| Hit rate | 80.5% | 84.99% | 86.73% | 86.43% |

2-2

Compared to direct-mapped or lower way\_number structure , using higer way\_number means the data will not be frequently modified and increase the probability the data in cache.

2\_3

Yes,hit rate keeps going higher. Because in higher way\_number, the cache will not modified the data frequently.

However,fully associative cache has become lower. I think that in fully associative cache , data will not be modified across to index,decrease the associativity between each data.

Q3



Mergesort is a [divide and conquer algorithm](https://en.wikipedia.org/wiki/Divide_and_conquer_algorithm).First it divides the unsorted list into *n* sublists, each containing 1 element (a list of 1 element is considered sorted) and then Repeatedly [merge](https://en.wikipedia.org/wiki/Merge_algorithm) sublists to produce new sorted sublists until there is only 1 sublist remain.

Quicksort is also a [divide and conquer algorithm](https://en.wikipedia.org/wiki/Divide_and_conquer_algorithm).First it picks an element, called a *pivot*, from the array.Second,it does *partition.* Reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. This is called the *partition* operation. [Recursively](https://en.wikipedia.org/wiki/Recursion_(computer_science)) apply the above steps to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values.

2.

Both of them are efficient algorithm.Average Time Complexity is Ο(n log n)

Space Complexity:

Quicksort: Ο(log n) ~ Ο(n)

Mergesort: Ο(n)

3.

Quicksort has the higher hit rate. Because quicksort change the array inplace. Cache benefits from multiple accesses to the same place in the memory, since only the first access needs to be actually taken from the memory , the rest of the accesses are taken from cache, which is much faster the access to memory.However, merge sort needs much more memory accesses . Since every accessory array you create is accessing the RAM again.

心得：

這次的作業在debug方面花了不少時間，尤其是在ＬＲＵ的部分，一開始沒有把counter寫到header file裡面，而是個別在read還有write的地方宣告，導致兩邊計算不同步，hit rate也超級低，但是寫完這次作業之後對cache的各種型態已經算蠻了解的了。