Report of CO_LAb6 Team34

● Detailed description of the implementation

1. Direct-mapped cache: 依照助教給的檔案，一行一行讀進來，但因為有些只有不到 8 個 bytes (32 bits)，所以要在前面補上 0，補滿 32 bits，接著要將十六進位改成二進位，以便接下來找出 index 和 tag。如果對應到的 index 的 valid 是 true 且 tag 和該 index 的 tag 也相等，那就代表 hit，所以 hit_num++，沒有 hit 的話，就將 tag 改為該 index 的 tag，最後再回傳 hit ratio 回 main，main 會利用 1 - hit ratio 來算出 miss ratio。

2. Set-associative cache: 我用一個 map 儲存整個 cache 的內容，前面存 index，後面存一個大小為 n 的 list。每當有新的 address 進來，先找她的 index 有沒有在 cache 裡面，如果沒有的話直接放到 list 以後加進去，如果有的話，檢查有沒有在 list 裡面，再檢查 list 長度。

	在 list 出現過	沒出現過
list.size()<n	前面刪掉、後面加新的	直接加新的
list.size()==n	前面刪掉、後面加新的	第一個刪掉、後面加新的

簡單來說，就是用 list 這個容器，依照時間先後順序儲存，實作 LRU。

● Implementation results

```
ruby@ruby-ubuntu:~/Computer-Organization/Cache_Simulator$ ./demo.sh
rm -f *.o
g++ -I./include main.cpp direct_mapped_cache.cpp set_associative_cache.cpp -o ma
in.o
============================ Direct mapped result ============================

 0.0795226  0.0660363  0.0547202  0.0553402  0.0920787 | 4096
 0.0624709   0.042784   0.031623  0.0244923  0.0398388 | 16384
 0.0570454  0.0356534  0.0234072  0.0159665  0.0124012 | 65536
 0.0565804  0.0350333  0.0227872  0.0151914  0.0114711 | 262144
------------------------------------------------------------
        16         32         64        128        256

====================== N-way set associative result ======================

Bloack size: 64

  0.110681   0.083553  0.0778174  0.0782824 | 1024
 0.0827779  0.0517749   0.041854  0.0398388 | 2048
 0.0547202  0.0362734  0.0306929  0.0280577 | 4096
 0.0403038  0.0297628  0.0266625  0.0244923 | 8192
  0.031623  0.0237172  0.0234072  0.0229422 | 16384
 0.0254224  0.0232522  0.0227872  0.0227872 | 32768
------------------------------------------
     1-way      2-way      4-way      8-way
```

● Problems encountered and solutions

(109550031 李旻融)

讀檔案的時候沒注意到長度不一樣，導致 stoi function 轉換錯誤，讀出一些奇怪的字元。後來在前面補零就解決了。

(109550031 紀竺均)

前面自作聰明把.h 檔改成.cpp 檔，然後在跑 function 一直出現問題，(一下子說沒有這個 module 一下子說我宣告兩次)，後來改回去就一切正常了....