快取行為模擬

計算機組織 LAB 5

王璽喆

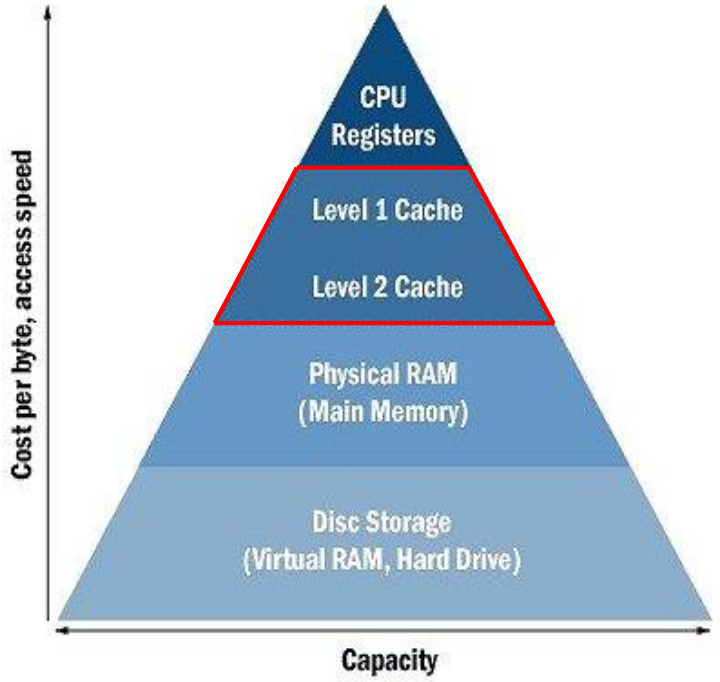
2018/12/4

OUTLINE

- Introduction of Cache
  - Cache Memory
  - Direct Mapped Cache
  - Associative Cache
  - Replace Policy
- 實驗目的
- Homework
- 評分標準

CACHE MEMORY

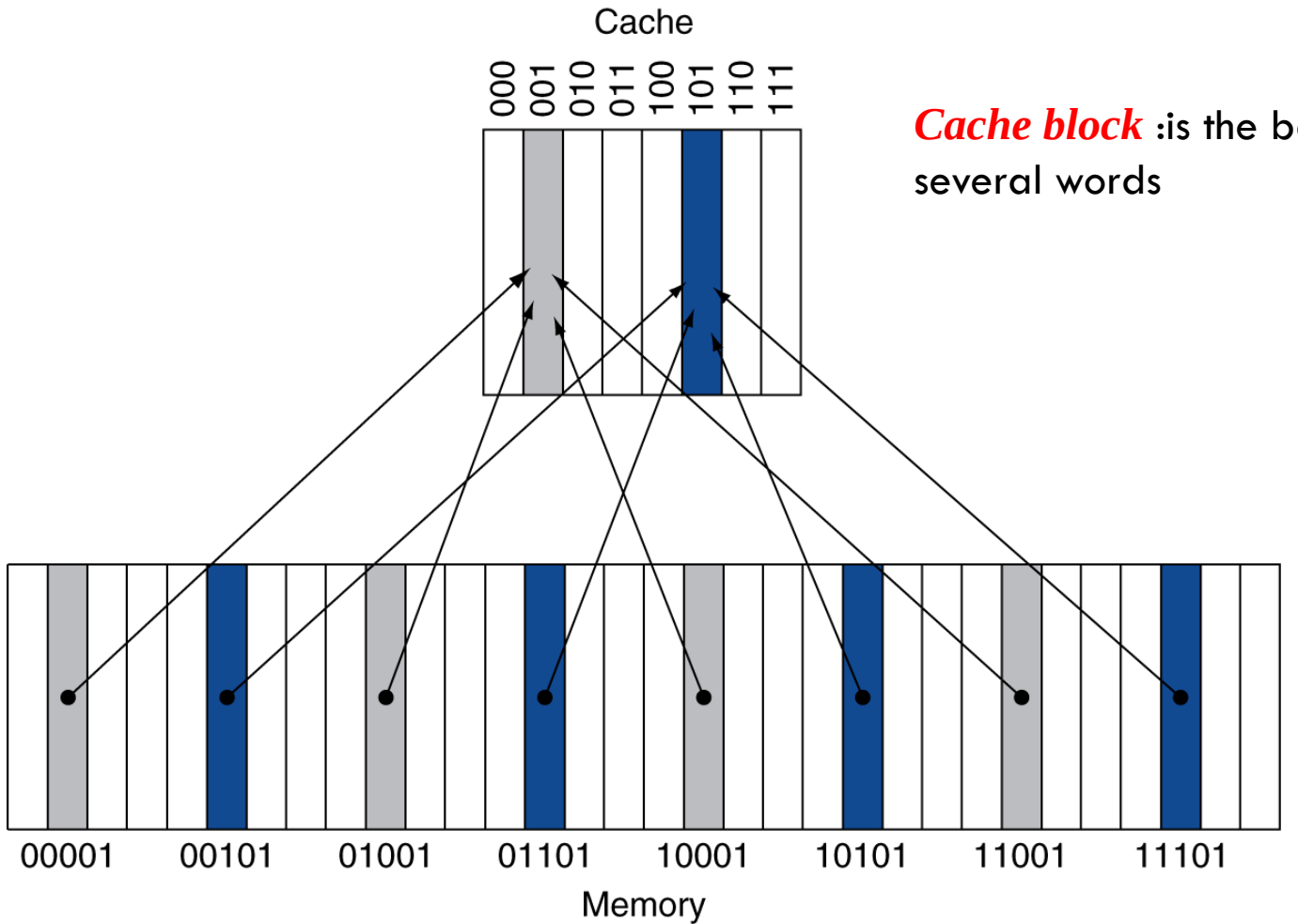
- Cache Memory
  - The level of the memory hierarchy closest to the CPU
- Computer Memory Architecture
  - 越往上層，記憶體的速度越快，容量越小。
  - 資料只能在相鄰的階層中移動。



computer memory architecture

DIRECT MAPPED CACHE

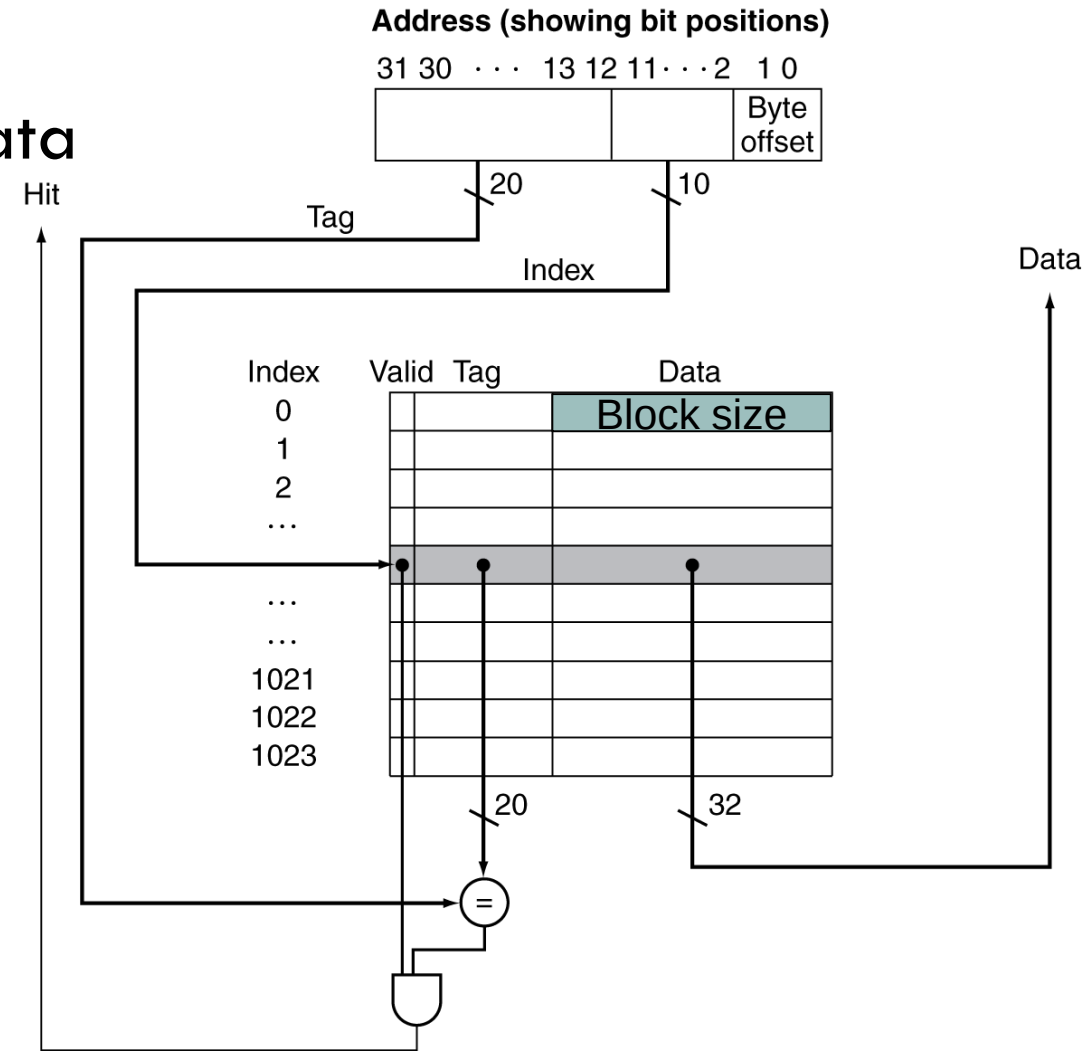
- 根據Memory位置，把所有區塊分配給cache。



Cache block :is the basic mapping unit, which usually contains several words

ADDRESS SUBDIVISION

- Tag
  - Store block address as well as the data
- Vaild bit
  - Cache中資料是否有效
- Block size
  - 1 words
- Cache size
  - 4 KB (1024 blocks)
    - 4KB =1Kwords = 1K blocks



DIRECT-MAPPED CACHE EXAMPLE (1/6)

- 8-blocks, 1 word/block, direct mapped

Initial state

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

DIRECT-MAPPED CACHE EXAMPLE (2/6)

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Miss	110

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	**Y**	**10**	**Mem[10110]**
111	N		

DIRECT-MAPPED CACHE EXAMPLE (3/6)

Word addr	Binary addr	Hit/miss	Cache block
26	11 010	Miss	010

Index	V	Tag	Data
000	N		
001	N		
010	**Y**	**11**	**Mem[11010]**
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

DIRECT-MAPPED CACHE EXAMPLE (4/6)

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Hit	110
26	11 010	Hit	010

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

DIRECT-MAPPED CACHE EXAMPLE (5/6)

Word addr	Binary addr	Hit/miss	Cache block
16	10 000	Miss	000
3	00 011	Miss	011
16	10 000	Hit	000

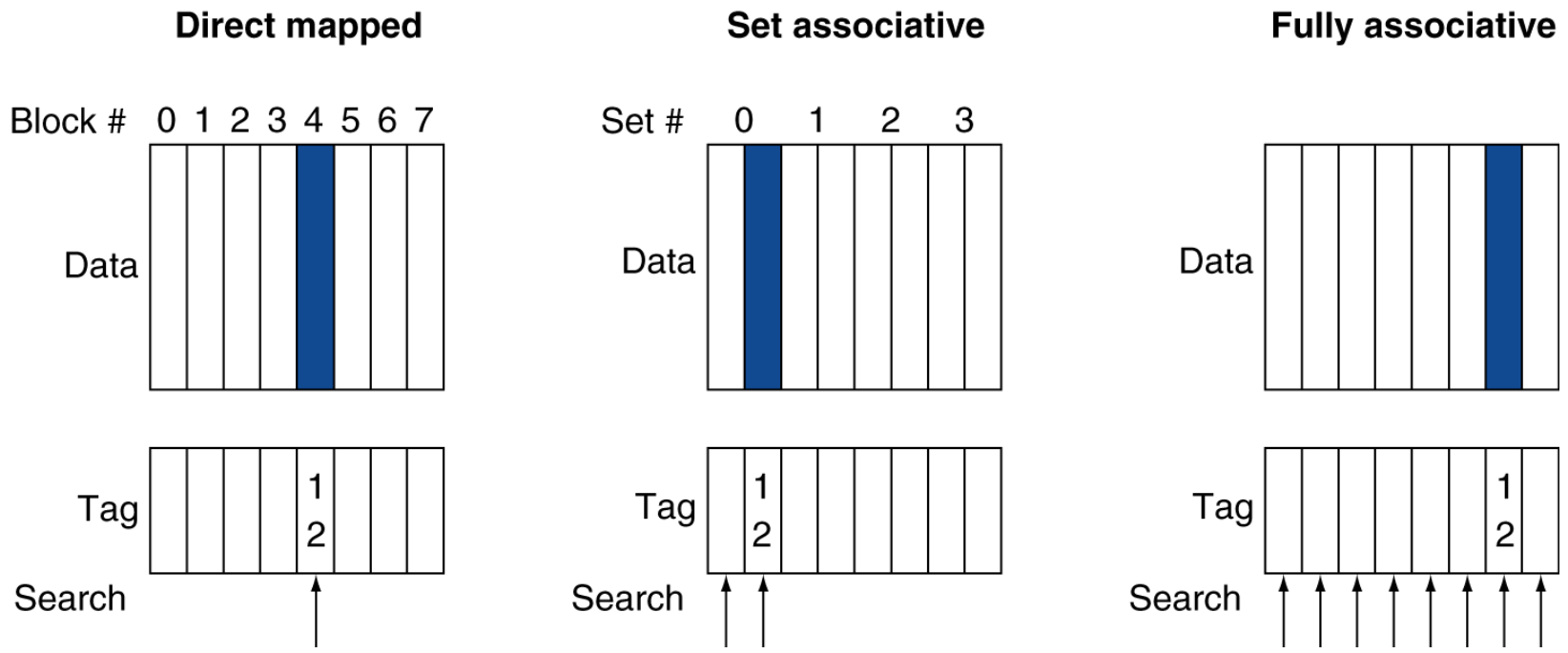
Index	V	Tag	Data
000	**Y**	**10**	**Mem[10000]**
001	N		
010	Y	11	Mem[11010]
011	**Y**	**00**	**Mem[00011]**
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

DIRECT-MAPPED CACHE EXAMPLE (6/6)

Word addr	Binary addr	Hit/miss	Cache block
18	10 010	Miss	010

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	**Y**	**10**	**Mem[10010]**
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

ASSOCIATIVE CACHE EXAMPLE (1/3)



ASSOCIATIVITY EXAMPLE (2/3)

Compare 4-block caches

- Direct mapped, 2-way set associative, fully associative
- Block access sequence: 0, 8, 0, 6, 8

Direct mapped

- Cache index = Block address % block numbers

Block address	Cache index	Hit/miss	Cache content after access				Cache index
			0	1	2	3	
0	0	miss	**Mem[0]**				
8	0	miss	**Mem[8]**				
0	0	miss	**Mem[0]**				
6	2	miss	Mem[0]		**Mem[6]**		
8	0	miss	**Mem[8]**		Mem[6]		

ASSOCIATIVITY EXAMPLE (3/3)

- 2-way set associative Block access sequence: 0, 8, 0, 6, 8

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	**Mem[0]**			
8	0	miss	Mem[0]	**Mem[8]**		
0	0	hit	**Mem[0]**	Mem[8]		
6	0	miss	Mem[0]	**Mem[6]**		
8	0	miss	**Mem[8]**	Mem[6]		

LRU(least recently used) Policy

Cache index = Block address % set numbers

- Fully associative Block access sequence: 0, 8, 0, 6, 8

Block address		Hit/miss	Cache content after access			
0		miss	**Mem[0]**			
8		miss	Mem[0]	**Mem[8]**		
0		hit	**Mem[0]**	Mem[8]		
6		miss	Mem[0]	Mem[8]	**Mem[6]**	
8		hit	Mem[0]	**Mem[8]**	Mem[6]	

實驗目的

- 以撰寫程式的方式來模擬cache行為，讓大家對cache更為熟悉。

HOMEWORK簡介

- LAB5壓縮檔內容
  - spice.din & gcc.din
    - 內含約十萬筆資料，以模擬CPU到cache找資料的行為
- 作業內容
  - 利用撰寫一個程式讀取din檔，來完成cache的行為模擬
    - 本次作業可以選擇使用C、C++來撰寫

作業 (OVERVIEW)

- CO_Lab5壓縮包裡面有.din檔
  - 裡面有資料 模擬CPU Cache行為時需要的指令資料
- 作業內容
  - 寫一個程式讀取並處理din檔，完成Cache的行為模擬
- 壓縮包
  - E-Course有放

作業 (執行檔)

把編譯完的執行檔命名為 cache

輸入語法:

cache cache_size block_size associativity replace_policy file

1. cache_size 8, 16, …, 256 (KB)
2. block_size 4, 8, 16, …, 128 (B)
3. associativity 1 (direct-mapped), 2, 4, 8, f (fully associative)
4. replace-policy FIFO, LRU
5. file 輸入檔名

輸入範例

cache 8 32 1 FIFO gcc.din

```
D:\GiantBaby\106-1course\CO\LAB\lab5\CO_Lab5>cache 8 32 1 FIFO gcc.din
Input file = gcc.din
Demand fetch = 1000002
Cache hit = 924129
Cache miss = 75873
Miss rate = 0.0759
Read data = 159631
Write data= 83030
Bytes from memory = 2427936
Bytes to memory = 352192

D:\GiantBaby\106-1course\CO\LAB\lab5\CO_Lab5>
```

作業 (.DIN檔內容)

.din檔內容格式

- Label
  - 0 : data read
  - 1 : data write
  - 2 : instruction
- Address
  - 由Tag、Index、Offset所組成且以16進位表示

Tag	Index	Offset

Lable

```
2 408ed4
0 10019d94
2 408ed8
1 10019d88
2 408edc
0 10013220
2 408ee0
2 408ee4
1 100230b8
2 408ee8
0 10013220
2 408eec
2 408ef0
2 408ef4
```

Addresss

作業 (輸出格式)

輸出要有這些項目(紅字為評分項目)

1. Input file
2. Demand fetch
3. Cache hit
4. Cache miss
5. Miss rate
6. Read data
7. Write data
8. Bytes from Memory
9. Byte to memory

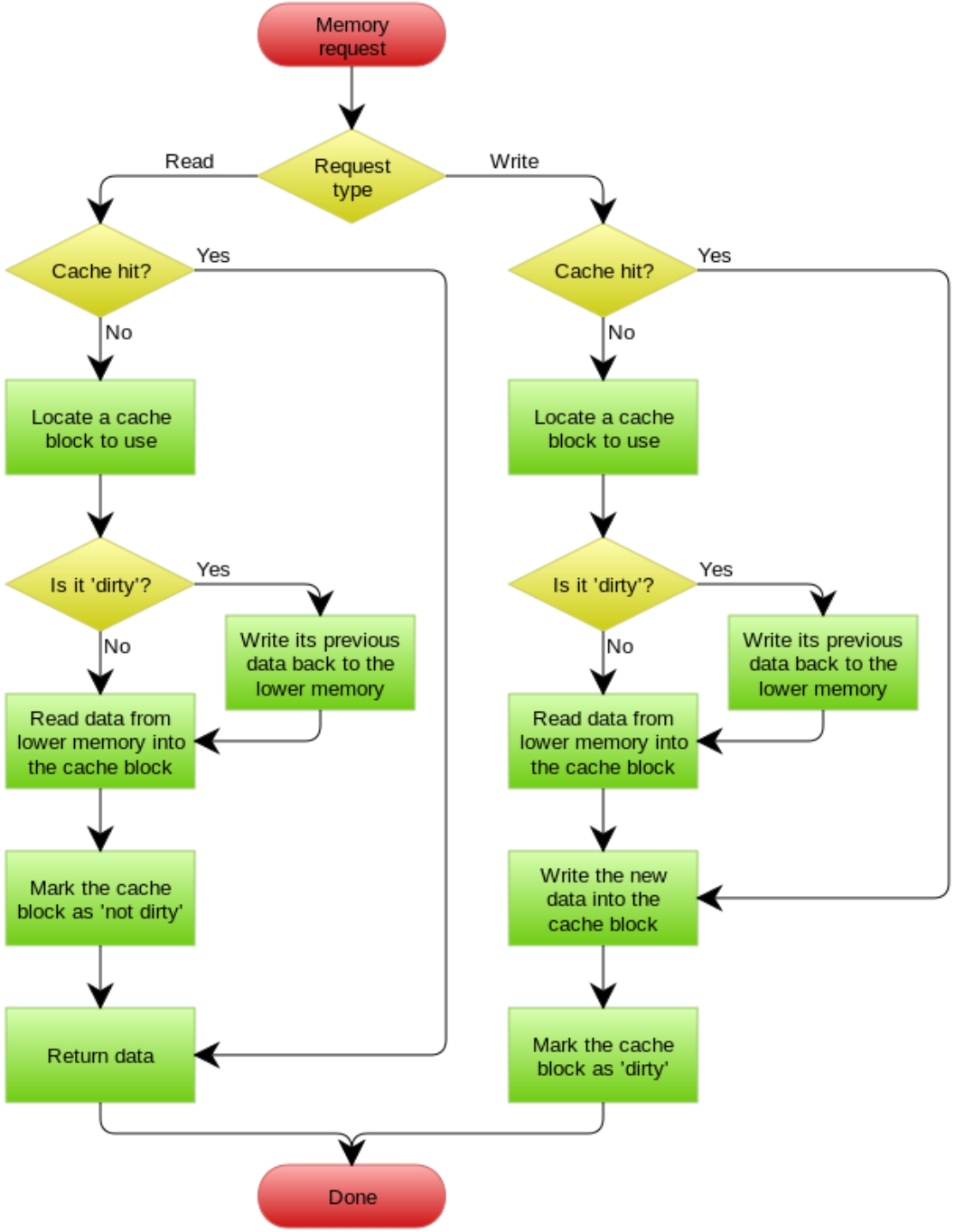
cache 8 32 2 LRU gcc.din

```
Input file = gcc.din
Demand fetch = 1000002
Cache hit = 940024
Cache miss = 59978
Miss rate = 0.0600
Read data = 159631
Write data= 83030
Bytes from memory = 1919296
Bytes to memory = 231424
```

cache 16 16 1 FIFO spice.din

```
Input file = spice.din
Demand fetch = 1000001
Cache hit = 970983
Cache miss = 29018
Miss rate = 0.0290
Read data = 150699
Write data= 66538
Bytes from memory = 464288
Bytes to memory = 71216
```

write-back 是將資料量儲存到一定的量之後,會依據同區塊的資料一次整批寫回去.裡面有提到 dirty ,他是在記憶體裡面 cache 的一個 bit 用來指示這筆資料已經被 CPU 修改過但是尚未回寫到儲存裝置中



作業 (評分標準)

作業總共100%

隨機設不同參數組合測試

檢查8個output項目個別是否正確(10%)

全對的話有另外有20分(20%)

QA中有附幾個組合以及其輸出結果。同學可以拿來測試程式執行結果是否與QA中附的執行結果一致。

作業 (繳交相關)

上傳有程式原始碼的壓縮檔到教學平台上

不要太奇怪的語言都可以(可用C, C++, Java, Basic, Python…)，盡量用C/C++，
非C/C++請附使用說明文件

Deadline = 12/19

END