Computer Architecture Project 2

 # Team Work:

MEMBER	B04705003 林子雋	B04902021 陳弘梵	B04902099 黃嵩仁
WORK	全部模組 debug、dcache memory 實作	整合 project1 模組、CPU.v 接線、report	實作 dcache_top 模組、memory 接線

 # How do you implement this project:

套用上一次作業的 module，在 pipeline 基礎下的，將 dcache 加入 EX/MEM 與 MEM/WB 之間，再設定各種 state 條件下去與 data_memory 溝通。

實作細項: (包含 # Cache controller in detail)

1. Cache 新增 stall 線路，連接到所有 latch 與 PC，當有 read、write 的 request，並且沒有 hit，會傳送 stall 訊號讓全部 latch 的輸出不變。
2. SRAM tag 模組負責比對 cache 資料是否 hit，hit 時會將值讀出並依照 offset 取出對應的 data 傳回 core 或是 data memory
3. Cache controller: 每次 clock 改動就檢查，初始化先將四個 flag 設為 0。

State	Idle		Miss		Read Miss	
Condition	Not Hit & get request	else	Has dirty bit set	No dirty bit set	Get memory ack	Not received ack yet
Mem enable	X	X	1	1	0	X
Mem write	X	X	1	0	1	X
Cache write enable	X	X	0	0	0	X
Mem write back	X	X	1	0	0	X
Change state	Miss	Idle	WriteBack	ReadMiss	ReadMiss OK	ReadMiss

此二表代表，再進入不同 state 情況下，切換到不同的 state 的順序，以及對於 flag 的改值方式，X 代表不做更動。

State	ReadMissOK	WriteBack	
		Get memory ack	Not received ack yet
Mem enable	0	0	X
Mem write	0	0	X
Cache write enable	0	1	X
Mem write back	0	0	X
Change state	Idle	ReadMiss	WriteBack

與 memory 溝通的部分，這四個 flag 的更動都直接輸出到 data memory 以及 dcache_data_sram 模組，依照著我們的設定，memory 也將在規定時間內返回 ack (分有 wait 與 idle state)。

 # problems and solutions

1. 這次的接線比上次簡單許多了，可以算是一下就完成，但是多了需要處理 state 的情況，之前都只有簡單的 if else 決定輸出。
2. 這次在實作之前，要先清楚想進入不同 state 的條件，以及進入不同 state 之後要採取的動作，否則很容易造成嚴重的錯誤。
3. 助教給的 Data_Memory.v 中沒有加 negedge rst_i 在 always block 中，因此造成一開始 state 會是 x，加上之後才修正。
4. 花了很多時間了解 state 的運作方式之後，撰寫速度就快了很多。
5. Write data 那個 block 花了很多時間 debug，因為 MIPS 是 byte addressable，但卻是一次要取 4 個 word(32bits)，所以花了不少時間在想那段要怎麼寫才是正確的。