Q: 當 core 1 & core 2 同時 store 到相同位址的 data memory，是否有優先順序? 如何決定其優先順序?

A : 比較兩個 core 分別執行到第幾""道""指令，ex. core 1 在執行第 20 道指令對 data memory 的 0x0000_1234 執行儲存的動作，而 core 2 在執行第 19 道指令對 data memory 的 0x0000_1234 執行儲存的動作，則 core 2 會先對 data memory 做存入的動作，接著才輪到 core 1，因此 jump or branch 不會造成影響，但兩個 core 完成的指令數要控制在一定範圍內，不然 data memory 的值有可能會亂掉。

Q: 是否可以認為當 stall ==0 for 1 cycle 就等於完成一條指令，interrupt 時則驗證完成 stall == 0 的 cycle 數的指令時 reg 與 data memory 的資料

A: 當 stall = 0 時，就表示 CPU 已完成該指令，此時助教會檢查對應 core 裡面所有 register 的值是否正確，例如: 當第一次 stall_core_1 拉為 0 時，助教就會知道你 core_1 處理完第一道指令，這時候會去檢查 core_1_r0~core_1_r7 的值是否正確。為了簡單化，interrupt_1 與 interrupt_2 只會各出現一次，且只會出現在最後助教要檢驗你們 data memory 時才會執行，例如: 今天助教想在 core_1 & core_2 各執行 1000 道指令後檢驗學生的 data memory，而當 core 1 完成 1000 道指令時，core 2 才執行到 990 道指令，此時 interrupt_1 會先被拉起來，等到 core 2 也執行 1000 道指令後開始計算 1500 個 cycle，助教會在此時檢驗 data memory，換言之，SRAM 內的值需要在這 1500 個 cycle 內存入 data memory。

Q: Final Project 的 CPU 是從什麼時候開始讀指令，且一次要處理幾個 instruction 呢?

A: 理應來說，reset 結束你們的 stall 都要為 high，因為你們尚未處理完第一筆指令，當你處理完時，stall 就會變為 low，此時助教會檢查你們內部每個 register 內部的值是否正確，而你的 stall 只會維持 1 個 cycle 的 low，直到你處理完第二筆指令時才會再度拉為 low。建議可以考慮使用 pipeline 架構，來提升 throughput，這部分可以參考 MIPS 架構如何做到這件事情。interrupt 訊號的用意在於打斷你 CPU 的工作，正常的 CPU 在處理 interrupt 都需要將目前的 program counter 指到的位置存起來，當 interrupt 結束後，CPU 就會繼續執行原本的程式。但這次 LAB 為了簡單化，只在助教想打斷你們程式的時候會執行，執行 interrupt 後即結束程式。如果沒有 interrupt 介入，CPU 會執行完 0x0000~0x2000 的所有指令，但考量到 demo 時間，助教會分別在 RTL、GATE、POST 時給予適時的 interrupt。

Q: Program counter 起始位置是多少? Instruction & data 在 memory 中是如何擺放?

A: Program counter 起始位置是 0x0000_0000。Instruction & data 在 memory 中擺放方式如下，instruction memory & data memory 每個位置的 size 皆為一個 byte (8-bits)，且採用 little endian 的方式存取