Computer Architecture Project 1

Member and Teamwork

B04902010 資工三 羅凱齡： hazard detection, wire connection
B04902013 資工三 鄧逸軒： forwarding
B04902116 資工三 蘇景耀： pipeline register, wire connection

Pipeline Impletation

藉由加入4條 pipeline registers (`IF_ID` , `ID_EX` , `EX_MEM` , `MEM_WB`) 以控制每個 cycle 每個 instruction 在兩條相鄰的 register 間執行需要的步驟；此時整個 CPU 每個 cycle 每個 unit 都會被使用，等於同時執行 5 個不同的 instructions (相較於 single-cycle CPU 而言) 。 在某些特定狀況下，forward 必要的資訊至前面的 stage 以維持 instruction 執行的正確性，也可能碰到無法解決的 hazard，必須加以判斷決定是否 stall。

Module Implementation

PC

由於需要在有 hazard 時 stall，PC 必須增加一個 input port `PC_write_i` 以決定是否 output 新的 PC，或者維持原本的輸出。

Registers

在 pipeline 實作下可能出現同一個 cycle 同時讀寫同一個 register 的狀況，因此在 Registers 必須實作簡單的 forwarding:

```
assign RSdata_o =
    (regwrite_i && (RSaddr_i==RDaddr_i) && RDaddr_i != 5'b0)
    ? RDdata_i : register[RSaddr_i];
assign RTdata_o =
    (regwrite_i && (RTaddr_i==RDaddr_i) && RDaddr_i != 5'b0)
    ? RDdata_i : register[RTaddr_i];
```

除了判斷 forwarding 之外，也特判掉 `RDaddr_i` 為 0 的情況（可能出現在 warm-start，亦即某些 pipeline stage 尚未收到任何有效 data，output 初始設定值時）

Control

只加入一些額外的 signal 來處理增加的 instrucions。值得一提的是，由於必須實作 pipeline 的關係，原本 signal 應接到各個相對應的 unit，此時大部份必須直接導向 `ID_EX` ，並繼續 propogate 到後面幾個 pipeline registers 中。

Pipeline Registers

增加四條 pipeline registers (`IF_ID` , `ID_EX` , `EX_MEM` , `MEM_WB`) 以存儲每個 stage 所需的資訊。需要注意的是有些 output port 必須提供初始值以避免其直接 output 出 don't care signal，造成後面出錯。

Forwarding

依照投影片上的敘述實作，將 `MEM_WB` 或 `EX_MEM` 的 data forward 到 `EX` stage。

Hazard Detection

判斷目前在 `ID_EX` register 中的 instruction 是否是 lw；若是，則判斷是否與 `IF_ID` stage 中需使用的 register 一致，決定是否要 stall 一個 cycle；stall 時必須往 `ID_EX` 發 NOP signal，同時 freeze PC 與 `IF_ID` 的 data。

Data Memory

大體上模仿 instruction memory 的寫法，需要注意的是 data memory 可能被寫入。

Others

基本沿用上一次作業的 code，並額外加入一些簡單的 components (MUX*, Shift2, ... e.t.c.)

Problems and Solutions

1. 命名不一致：多討論，統一命名規則。
2. wire / register 為 x(don't care)：設定初始值，或者特判之。
3. pipeline 超前：用 gtkwave 找出超前的 stage，重新設定 wire connection。