計算機組織 Lab5

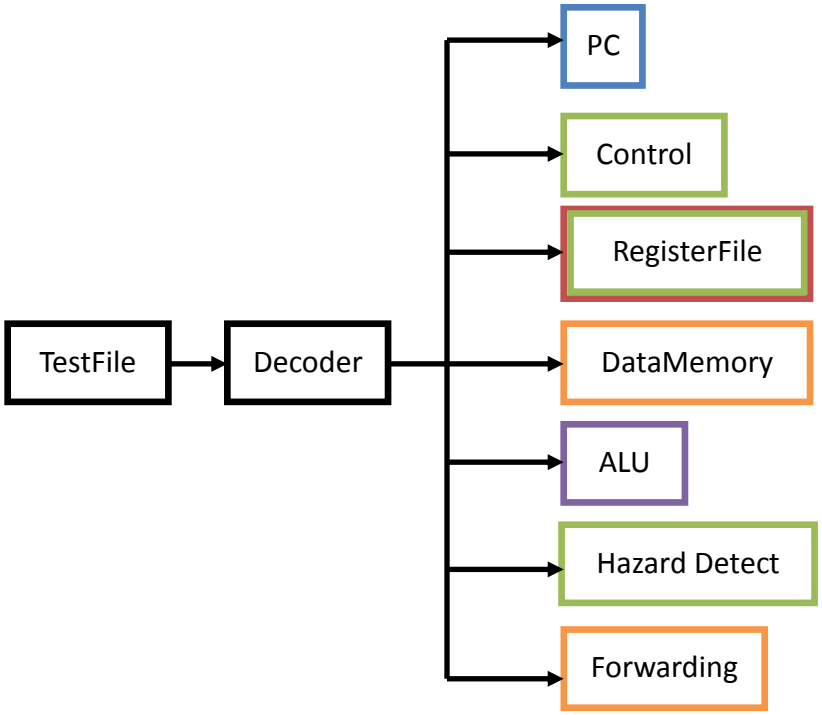
9617145 許晏峻

各元件功能：

每個 block 都代表一個獨立的 module：

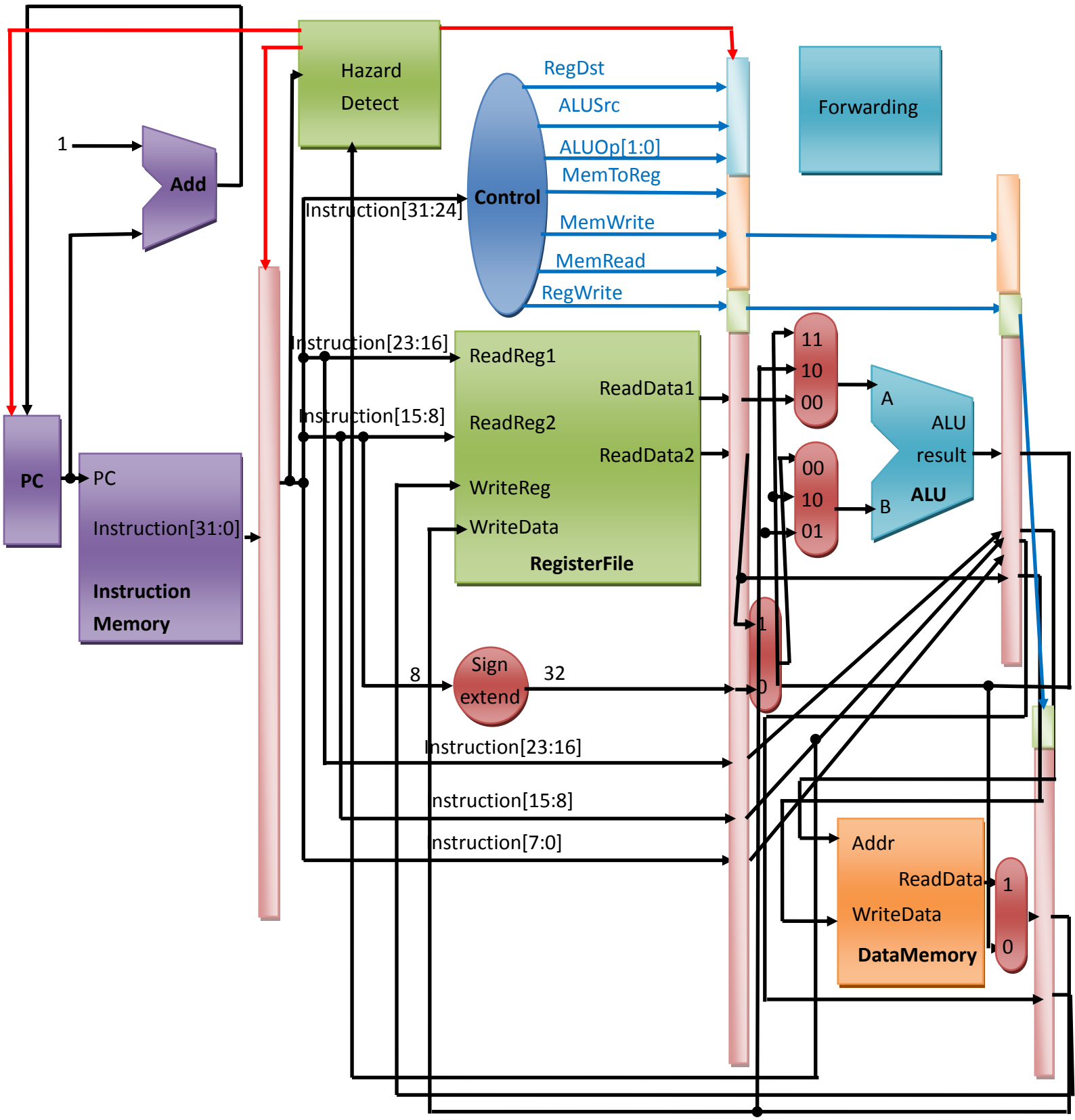
1. 由 TestFile 開始整個指令，作 Instruction Memory 的初始化
2. PC 則依照 PC 值取出 InstructionMemory 傳進 Decoder
3. Control 負責的是所有的 control signal for MUX
4. RegisterFile 做 register 的存值和讀值，存值是在 positive clock 進行
5. DataMemory 做 memory 的存值和讀值，存值是在 positive clock 進行
6. ALU 做各種運算包括 add、sub、shift left、shift right
7. Decoder 做各種傳值的動作，並且做各個 state register 的賦值
8. HarzardDetect 從 Decoder 拿 state register 作 hazard 的偵測
9. Forwarding 從 Decoder 拿 state register 來作 forwarding

因此在 Verilog 中的架構如下圖：
每個 module 都是 link 到 Decoder，Decoder 設定 state register

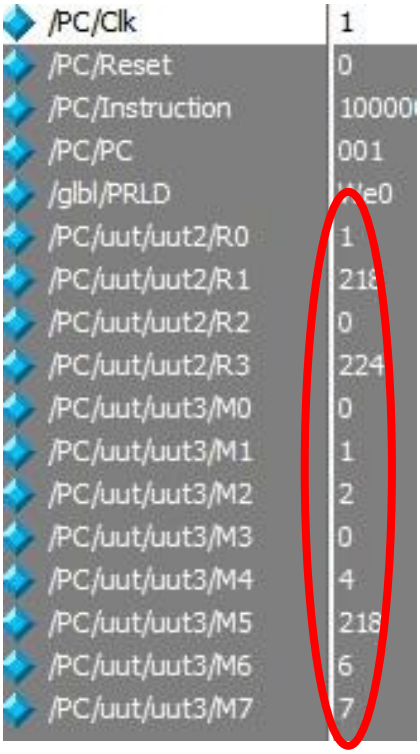


架構圖：

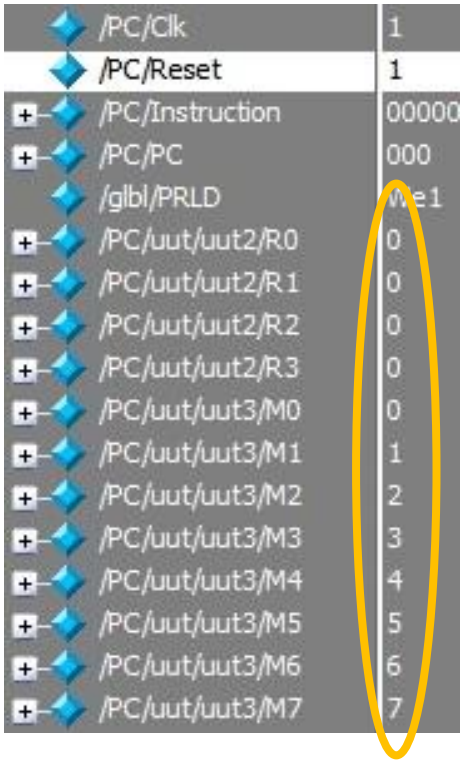
在整個 simgle-cycle 的概念上架構如下圖：
我的設計是在 positive edge 來作 write 的動作，因此 write 會在下個 clock 來的瞬間做好
以下將每個 module 做的事用顏色區隔開來



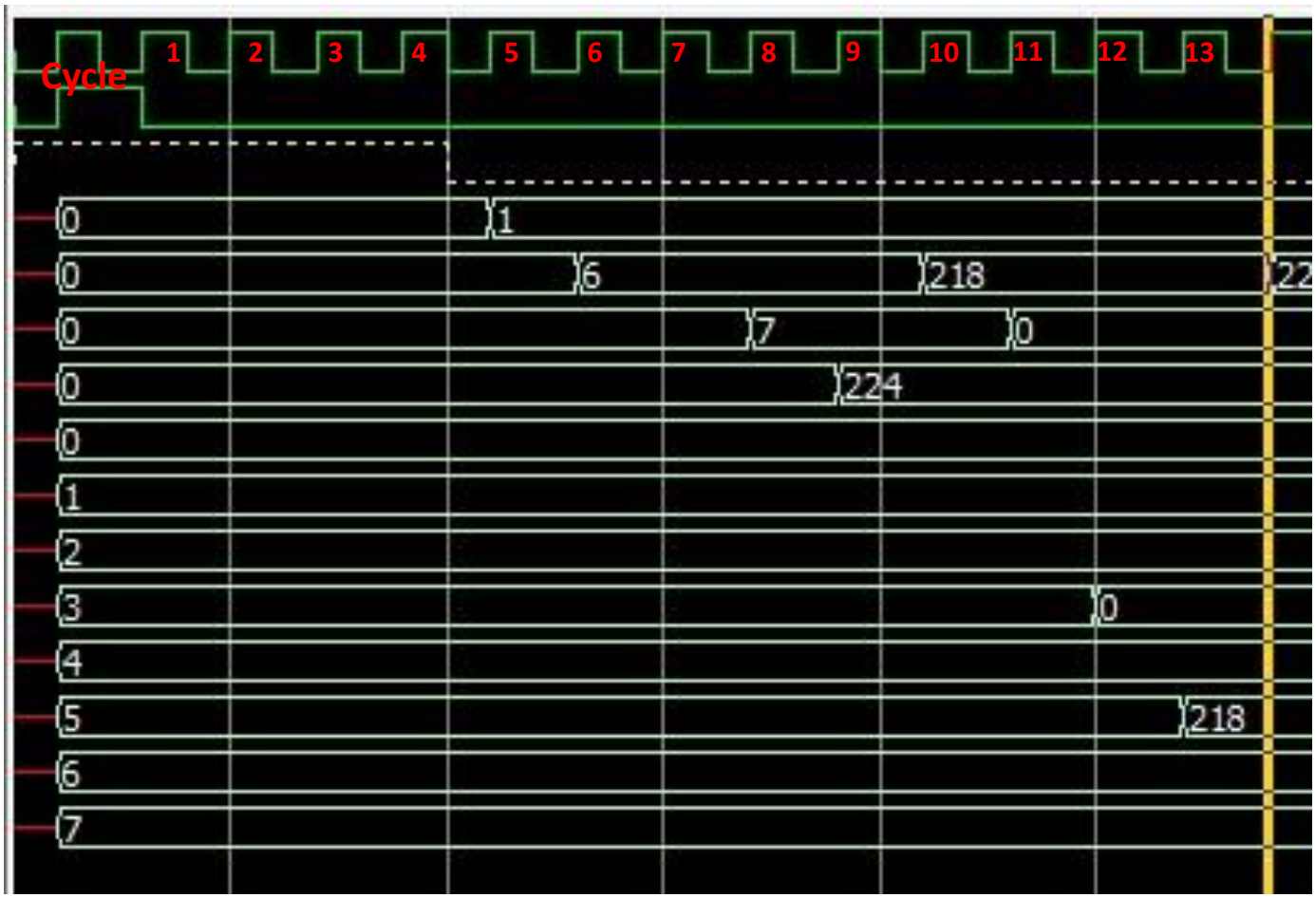
波型圖：



Final Vlaue



Initial Vlaue



Cycle 分析:

Cycle1:

IF: 讀指令LOAD M1 R0 (LOAD M1 R0)

Cycle2:

IF: 讀指令LOAD M6 R1(LOAD M6 R1)

ID: nothing(LOAD M1 R0)

Cycle3:

IF: 讀指令ADD R0 R1 R2(ADD R0 R1 R2)

ID: nothing(LOAD M6 R1)

EX: nothing(LOAD M1 R0)

Cycle4:

IF: 讀指令SL R2 5 R3(SL R2 5 R3)

ID: 讀出 R0、R1，偵測到 Hazard(ADD R0 R1 R2)

EX: nothing(LOAD M6 R1)

MEM: 讀出 M1(LOAD M1 R0)

Cycle5:

IF 讀指令 SL R2 5 R3(SL R2 5 R3)

ID: 讀出 R0、R1(ADD R0 R1 R2)

EX: (STALL)

MEM: 讀出 M6(LOAD M6 R1)

WB: M1 寫回 R0(LOAD M1 R0)

Cycle6:

IF: 讀指令SUB R3 R1 R1(SUB R3 R1 R1)

ID: 讀出 R2(SL R2 5 R3)

EX: 從 WB 做 forwarding 到 R1，R0+R1(ADD R0 R1 R2)

MEM: (STALL)

WB: M6 寫回 R1(LOAD M6 R1)

Cycle7:

IF: 讀指令SR R0 2 R2(SR R0 2 R2)

ID: 讀出 R3、R1(SUB R3 R1 R1)

EX: 從 MEM 做 forwarding 到 R2，R2<<5(SL R2 5 R3)

MEM: nothing(ADD R0 R1 R2)

WB: (STALL)

Cycle8:

IF: 讀指令STORE M3 R2(STORE M3 R2)

ID: 讀出 R0(SR R0 2 R2)

EX: 從 MEM 做 forwarding 到 R3，R3-R1(SUB R3 R1 R1)

MEM: nothing(SL R2 5 R3)

WB: R0+R1 寫回 R1(ADD R0 R1 R2)

Cycle9:

IF: 讀指令STORE M5 R1(STORE M5 R1)

ID: 讀出R2(STORE M3 R2)

EX: R0>>2(SR R0 2 R2)

MEM: nothing(SUB R3 R1 R1)

WB: R2<<5 寫回 R3(SL R2 5 R3)

Cycle10:

ID: 讀出R1(STORE M5 R1)

EX: nothing(STORE M3 R2)

MEM: nothing(SR R0 2 R2)

WB: R3-R1 寫回 R1(SUB R3 R1 R1)

Cycle11:

EX: nothing(STORE M5 R1)

MEM: 從 WB 做 forwarding 到 R2，R2 寫回 M3(STORE M3 R2)

WB: R0>>2 寫回 R2(SR R0 2 R2)

Cycle12:

MEM: R1 寫回 M5(STORE M5 R1)

WB: nothing(STORE M3 R2)

Cycle13:

WB: nothing(STORE M5 R1)

心得：

這是目前做複雜的作業了，除了要把指令切成很多個 cycle 做之外，還要去設定許多 state register，還有 Hazard 的 detection 和 Forwarding 要做。

運用上課時教的概念，在 decoder 中宣告了許多 state register 給每個對應的 state，然後傳進 RegisterFile、DataMemory、ALU、Control、HazardDetect、Forwarding，就能分別以不同的 state register 來做不同的指令。