工作分配:

柏序: 70%

郁文: 20%

奕均: 10%

Control Hazard

ID::ID\_beq (Beq)

為beq instruction在decode的時候就檢查有沒有equal(提早檢查可以少flush一個instruction), 這個模組的output port Beq (1 bit) 會在equal的時候是1

我們把這個output bit接到Control的input port Beq, 當它是1, Control的output port flush\_op和PC\_MUX\_op才會被set

再把flush\_op這個signal接到Buf\_IF\_ID, 如果這個signal是1, IF/ID pipeline register就會清空, 原先預處理的指令就消失了

至於下一輪PC要有正確的值, 這點由PC\_MUX\_op處理, 這個output port會接到PC\_Mux模組; 當PC\_MUX\_op有set, PC才會跳到適當的位置，而不是像平常一樣加四。

Data Hazard

還沒完全研究出來 求解><

Data Forwarding

還沒完全研究出來 求解><

Pipeline

由Control開始, input port: instruction和Beq, Beq處理當beq指令成立的時候要做的事(Control Hazard), instruction負責產生output port Opcode, Valid

Opcode的用意是將opcode在各pipeline register中傳遞, 等到需要用到任何Control Signal的時候再即時decode, 又發現這次project的Opcode只有[6:4] bit這三個bit相異, 因此只要傳這三個bit即可

至於valid是用來檢查指令是否有效，當opcode在project1範圍外, valid bit就會是0 或當IF/ID pipeline register 被清空時, 會讀到指令32’b0, 也會有這種情形

傳遞情形:

opcode\_ID => opcode\_EX => opcode\_MEM => opcode\_WB

opcode\_ID -> Branch Signal (Beq\_Op in our code)

opcode\_EX-> ALUOp, ALUSrc Signal

opcode\_MEM -> MemWrite Signal (mem\_write in our code)

opcode\_WB-> RegWrite Signal (regwrite\_MEM, regwrite\_WB in our code)

opcode\_WB-> MemtoReg Signal (reg\_src\_WB in our code)