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)