**Computer Organization**

**HDL simulator you used (ModelSim or Xilinx):**

**ModelSim**

**The input fields of each pipeline register:**

**IFID Reg: clk\_i, rst\_n, PC\_Plus4, Inst, InstReg, PC\_Plus4Reg**

**IDEX Reg: clk\_i, rst\_n, EX(Regdst,** **ALUSrc, ALUOp, Branch\_type\_select), MEM(Branch, Memread, Memwrite), WB(MemtoReg, Regwrite), RsData , RtData, imm\_value, zero\_value, Shifter, RegRt, RegRd, func\_i**

**EMMEM Reg: clk\_i, rst\_n , WB, M, PC\_Branch, ALUOut, RegRD, Mem\_WriteData, Branch\_type\_out, WBreg, Mreg, PC\_Branchreg, ALUreg, RegRDreg, Mem\_WriteDatareg, Branch\_typereg**

**MEMWB Reg: clk\_i, rst\_n, WB, Memout, ALUOut, RegRD, WBreg, Memreg, ALUreg, RegRDreg**

**Compared with lab4, the extra modules:**

**ADD: Four pipeline modules**

**Delete: Jump\_choose modules**

**Problems you met and solutions:**

**一開始看到pipeline一直在想要如何做到在一個cycle裡面同時做pipeline讀寫並且在每一個階段正確執行，後來仔細研究一下code，發現是透過posedge 和 negedge來達到這樣的效果。另外在做CPU兜線的時候，因為是從Simple cycle直接轉換過來，整體架構其實都不太一樣，因此再做之前，必須事先把所有wire重新規劃一遍，並且分清楚每一個stage需要拉的wire，是一個非常需要細心的lab。**

**Summary:**

**CPU在升級pipeline之後，要注意的事情變得更複雜了，雖然前面simple cycle都有完善的註解與安排，但是在pipeline最大的問題還是hazard的判斷與拉線，有些地方的架構甚至需要重新做一些安排，希望接下來在做lab5的時候能有更詳細更完善的提示。**