IF ID EXE MEM WB

IF/ID ID/EX EX/MEM MEM/WB

rd_data_ME FWJalrSel rs1_data_ID forwardA adder imm_out_ID immPlusRs1_ID PC_next

PC_plus4_IF PC_plusImm_ID PCAorPCimm_ID Jalr_ID

(branch_flag_ID | jal_ID) & Branch_ID

PC_plus4_IF 4 PC_IF adder PC_plus4_ID

PC_ID imm_out_ID adder PC_plusImm_ID imm_out_ID

wdPC_ID rdPCSel_ID

rdPC_EX rdPC_ME

aluResult_ME rdPC_ME CSR_out_ME MemtoReg_ME rd_data_ME rd_data_WB

LoadData_MEM

PC_IF PC_ID

IM addr O Inst_IF

Inst_ID

RF rs1_addr_ID rs2_addr_ID rd_data RegWrite. rs1_data_ID rs2_data_ID

rd_data_ME FWCompSrcASel comparA Comparator comparB FWCompSrcBSel rd_data_ME funct3_ID branch_flag_ID

Controller opcode_ID funct3_ID

RegWrite_ID rs2Sel_ID rdPCsel_ID MemWrite_ID MemtoReg_ID MemRead_ID Branch_ID Jal_ID Jalr_ID ALUop_ID

RegWrite_EX rs2Sel_EX MemtoReg_EX MemRead_EX ALUop_EX

RegWrite_MEM RegWrite_WB

MemWrite_MEM MemtoReg_ME MemRead_MEM MemWrite_EX

rs1_data_EX rd_data_ME rd_data_WB FWSrcASel ALUsrcA_EX ALUresult_EX src_A result ALU src_B ALUsign ALUctrl

rs2_data_EX rd_data_ME rd_data_WB FWSrcBSel ALUsrcB_EX rs2orForward rs2_data_EX ALUctrl_EX

ALUresult_EX ALUresult_MEM addr O DM data MemWrite MemRead

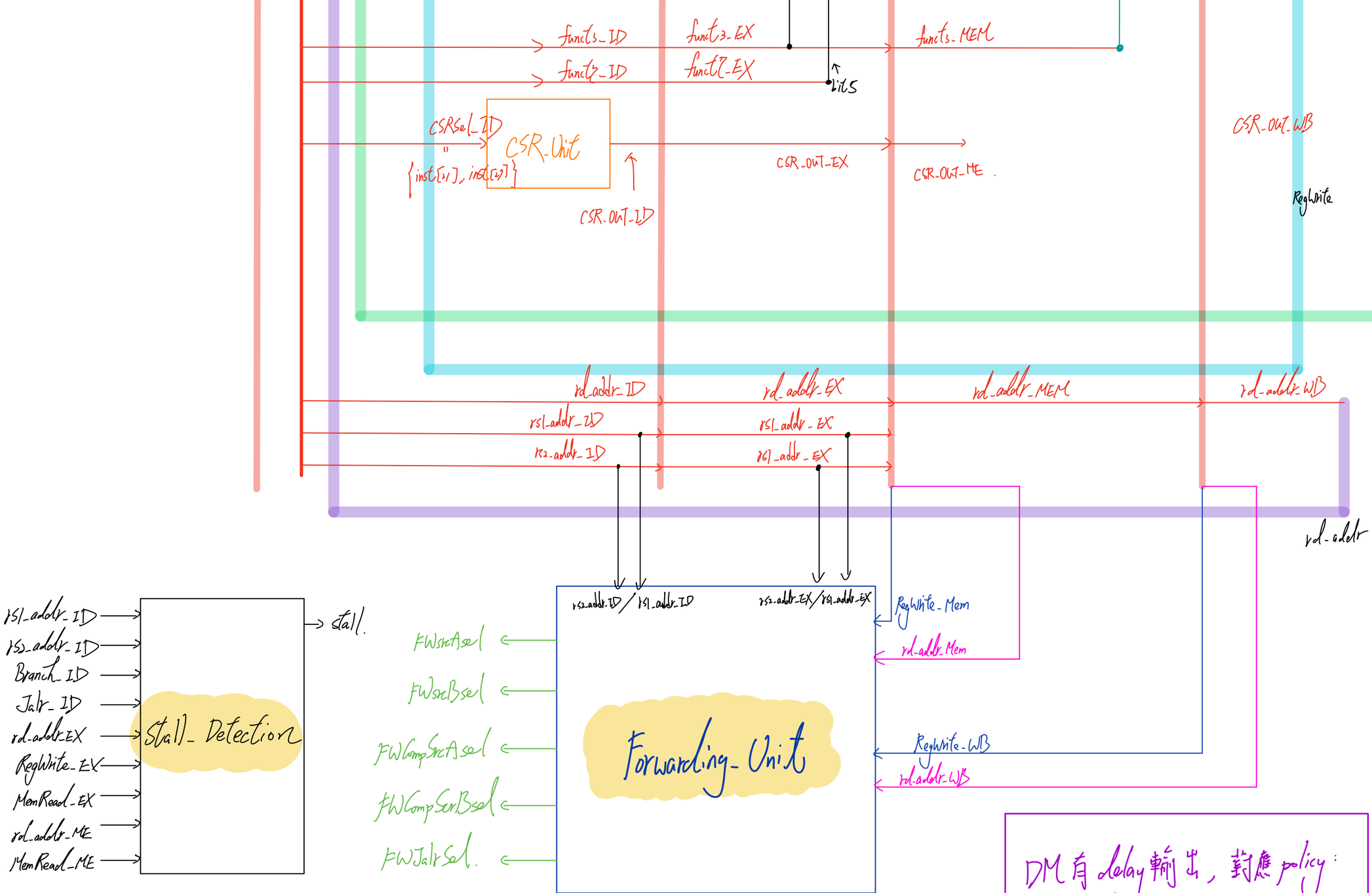
LoadData Mask funct3_MEM

ALUresult_MEM[1:0] ALUresult[1:0] wren_in wren_out Alignment Correction wdata_in wdata_out DM_WEB DM_DI

rs2_data_MEM rs2orForward rs2_forward

Inst_ID Imm_Gen imm_out_ID imm_out_EX

ALU Controller funct3 funct7_6 ALUop_EX



Note: 若 WB stage 要寫回 rd 時，剛好有指令 read rd，則當來不及，可以讓 RF 的輸出=輸入來解決須等待一 cycle 的問題！

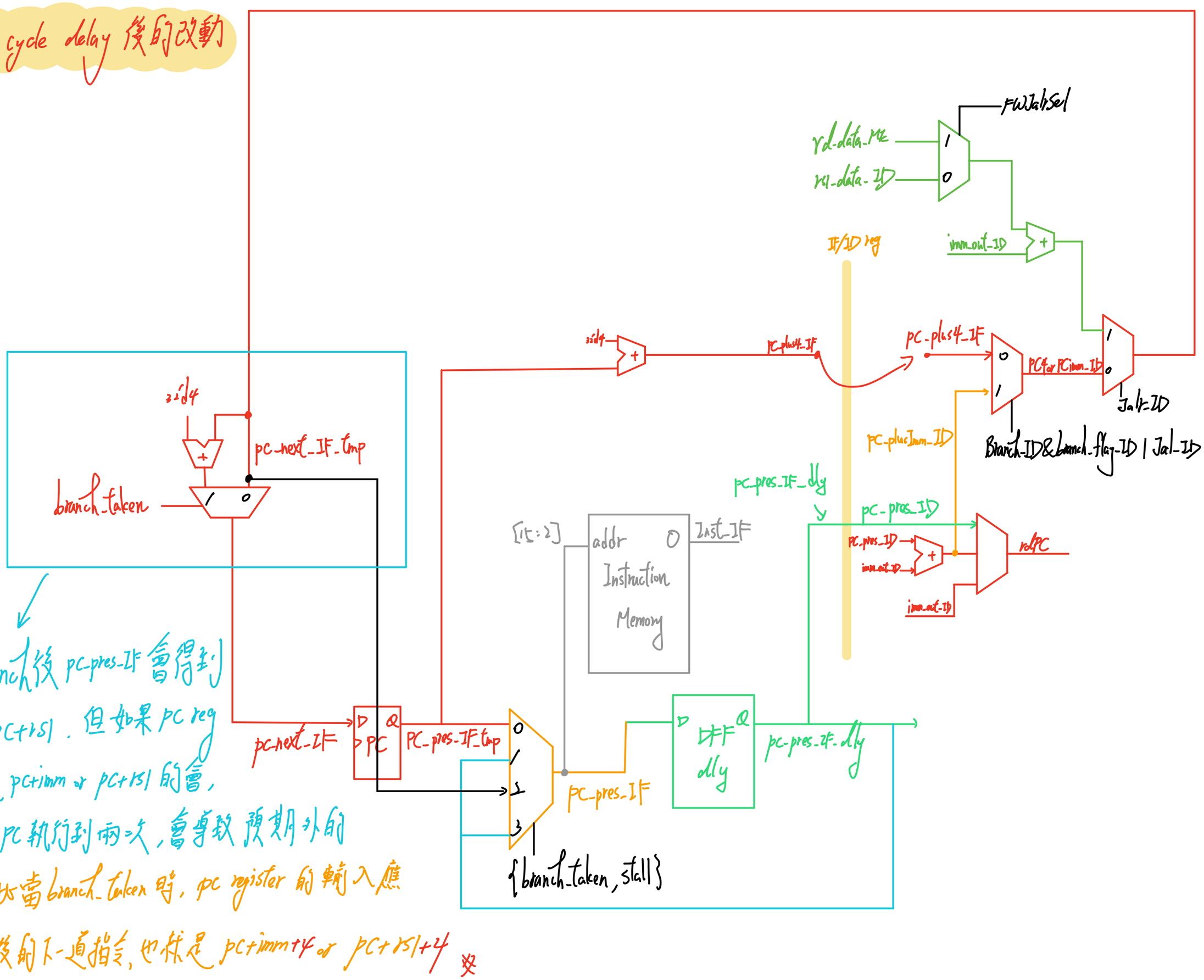
DM 有 delay 輸出，對應 policy：

1: 將 addr 提早一個 cycle 給，data 搶贏須 delay 的 DM 同時出來。

2: 因為會影響到寫入時序，因此也將寫入 data & write enable 提早給。

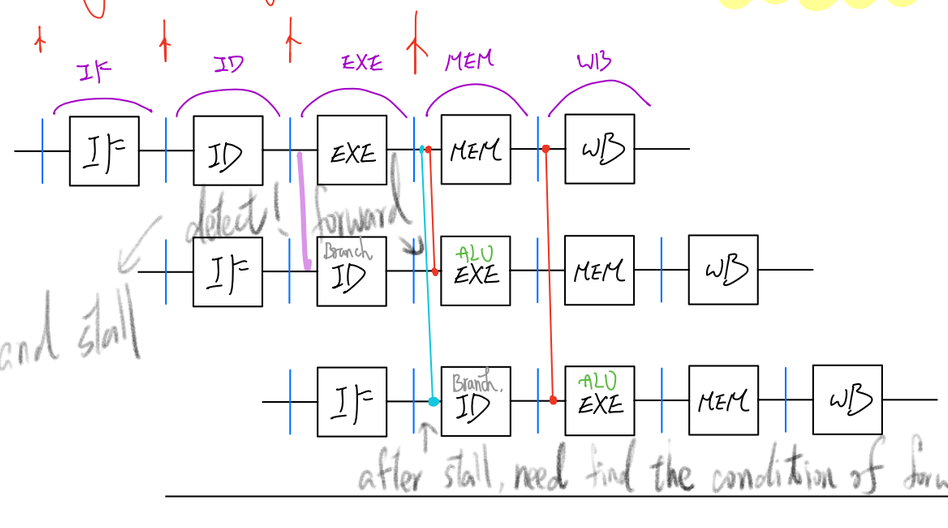
rs2orForward　MemWrite_EX

IM有 one cycle delay 後的改動



由於在branch後 pc_pres_IF 會得到 pc+imm or pc+rs1. 但如果 PC reg 的 input也是 pc+imm or pc+rs1 的會, 會使同一個PC執行到兩次, 會導致預期外的錯誤, 因此當branch_taken時, pc register 的輸入應為 branch 後的下一道指令, 也就是 pc+imm+4 or pc+rs1+4 ※

Only forwarding



R-R or R-X-R

```
if(RegWrite_MEM==1 && rd_addr_MEM≠0)
    if(rd_addr_MEM == rs1_addr_EX) ALUforwarding_srcA = ALUresult_MEM   FWsrcAsel = 2'b01
    if(rd_addr_MEM == rs2_addr_EX) ALUforwarding_srcB = ALUresult_MEM   FWsrcBsel = 2'b01
else if(RegWrite_WB==1 && rd_addr_WB≠0)
    if(rd_addr_WB == rs1_addr_EX) ALUforwarding_srcA = ALUresult_WB     FWsrcAsel = 2'b10
    if(rd_addr_WB == rs2_addr_EX) ALUforwarding_srcB = ALUresult_WB     FWsrcBsel = 2'b10
```

R-Branch

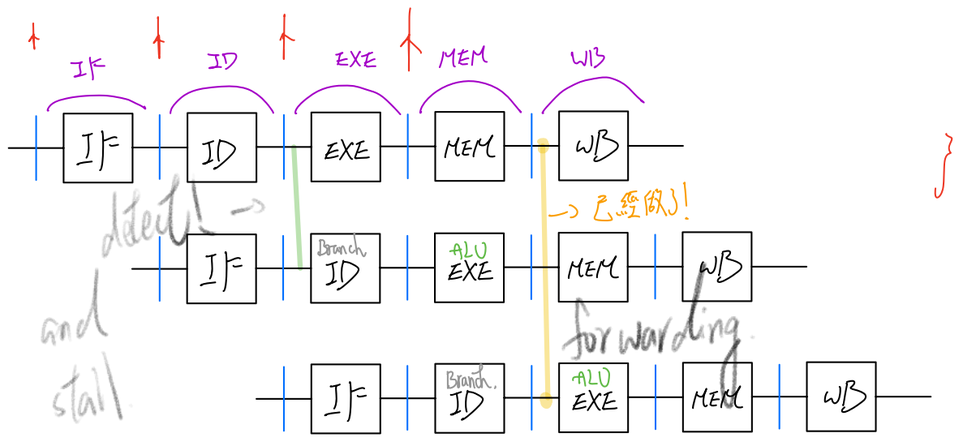
need to stall for one cycle.
mind the flush!!

```
if(RegWrite_MEM==1 && rd_addr_MEM≠0) {
    if(rd_addr_MEM == rs1_addr_ID)  FWCompSrcAsel = ALUresult_MEM
    if(rd_addr_MEM == rs2_addr_ID)  FWCompSrcBsel = ALUresult_MEM
}
```

```
if(regWrite_EX==1 && (Branch_ID | Jalr_ID)) {        // 因為ID才decode出來，所以上一條用EX去偵.
    if(rd_addr_EX== rs1_addr_ID || rd_addr_EX == rs2_addr_ID)
        stall=1
    else
        stall=0.
}
```

stall 1 cycle & forwarding

Load-R => need stall one cycle.

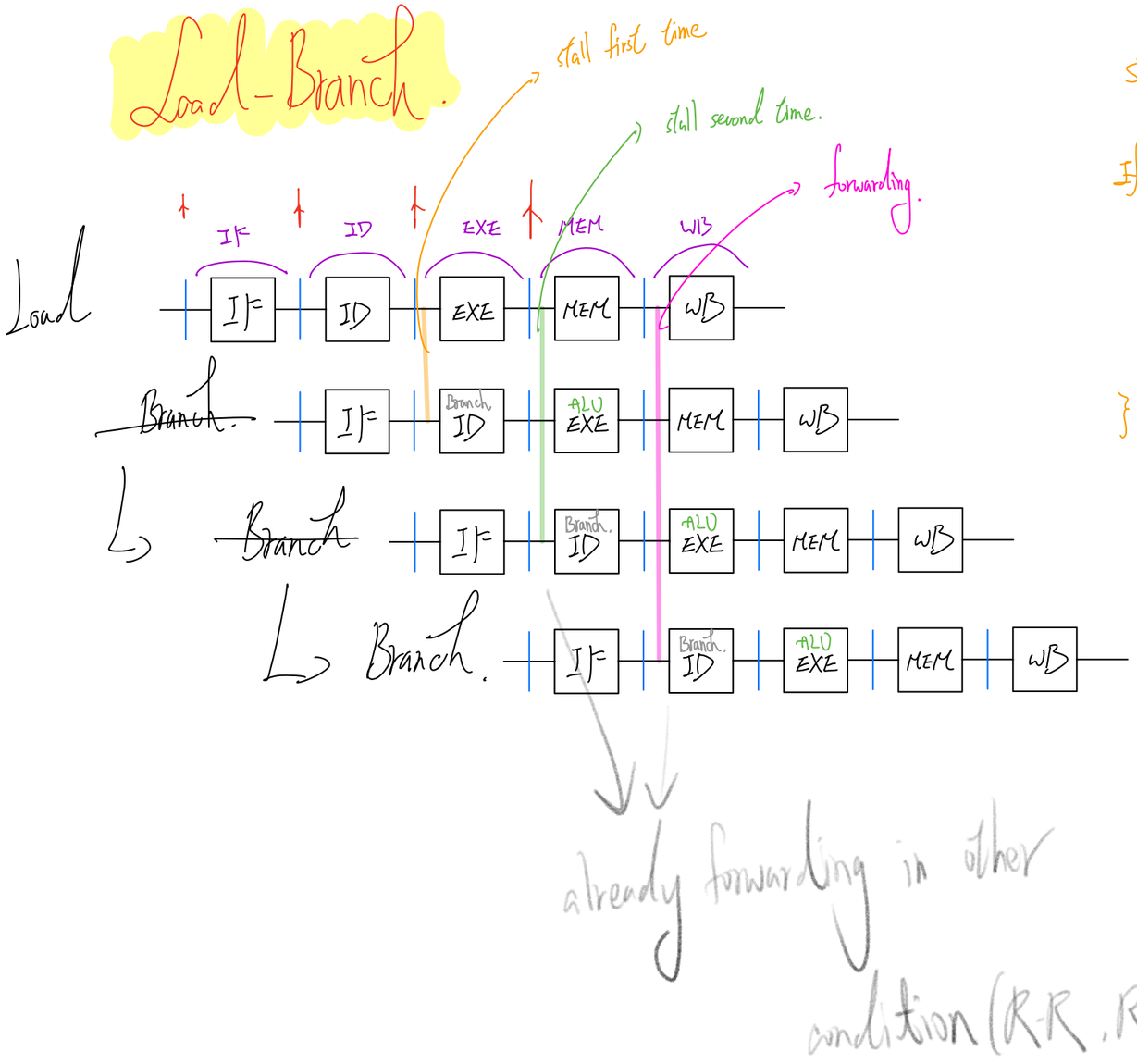



```
if(MemRead_EX==1 && RegWrite_EX==1) {        (X)
    if(rd_addr_EX == rs1_addr_ID || rd_addr_EX == rs2_addr_ID)
        stall=1
    else
        stall=0
}
```

Same.

stall 1 cycle & forwarding.

Load-Branch.



stall first time. (S1)

```
If(memRead_EX==1 && RegWrite_EX==1){
    if(rd_addr_EX == rs1_addr_ID || rd_addr_EX == rs2_addr_ID)
        stall=1
    else
        stall =0
}
```

(S2)

stall second time.

```
If(memRead_MEM==1 && (Branch_ID | Jalr_ID)) {
    if(rd_addr_MEM = rs1_addr_ID || rd_addr_MEM==rs2_addr_ID)
        stall=1;
    else
        stall =0;
}
```