這次有遇到幾個問題

第一個是 case 沒有寫完整，原本少了 default 導致 PC 一直是 xxxx，讓我們 debug 了超久。

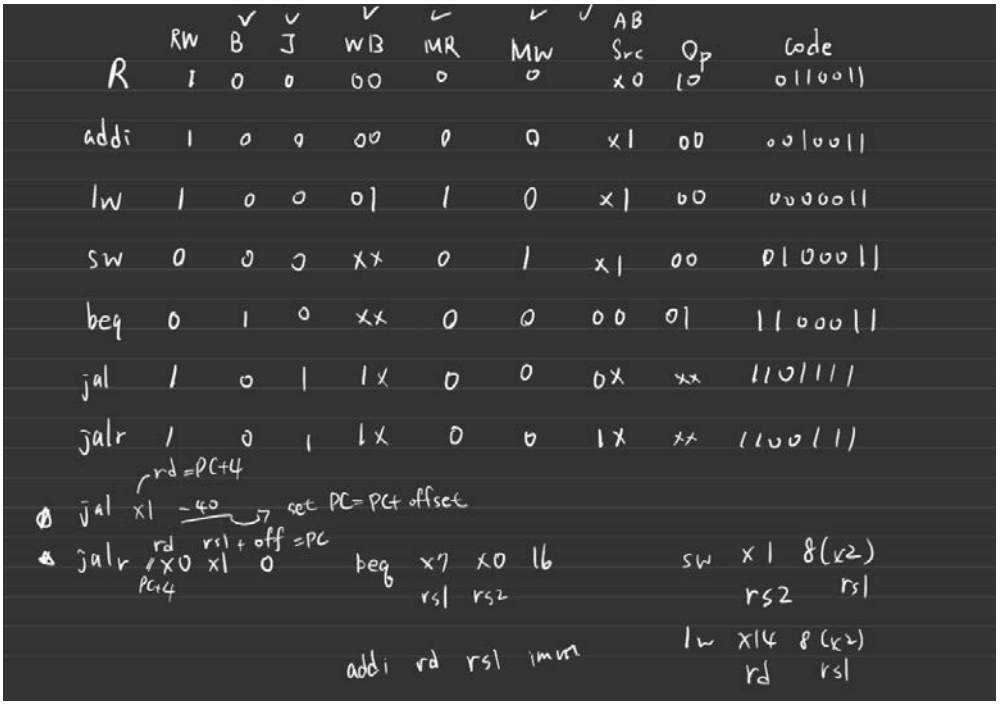
第二個就是 alu 的 input 要加 signed，我們覺得可能是因為 alu 後來是直接改成用對應的 operator 去實作，所以若是 input 沒有 signed 的話他會報錯。由於第 1700 行的指令是 slt，然後好像是 0 跟-1 比，最後結果應該要是 1，後來是在 simple_cycle 裡面，Vscode 會提示各個 port 的型態，才突然想到 alu 的 input 應該要是 signed，後來改完就對了~

第三個就是我們以為 jalr 是 Jtype，所以原本 Imm-Gen 有錯，後來改成 Itype 的就好了。

執行結果：

```
kkmelon@DESKTOP-TEEM7HG:~$ cd ComputerOrganization/LAB4/
kkmelon@DESKTOP-TEEM7HG:~/ComputerOrganization/LAB4$ ls
ALU_Ctrl.v     Decoder.v       Instruction.txt  MUX_2to1.v       Reg_File.v          alu.v       result.txt
Adder.v        Imm_Gen.v       Lab4-Slide.pdf   Makefile         Simple_CPU.lxt      answer.txt  test
Data_Memory.v  Instr_Memory.v  Lab4-Spec.pdf    ProgramCounter.v  Simple_Single_CPU.v  demo.sh     testbench.v
kkmelon@DESKTOP-TEEM7HG:~/ComputerOrganization/LAB4$ chmod +x demo.sh
kkmelon@DESKTOP-TEEM7HG:~/ComputerOrganization/LAB4$ ./demo.sh
iverilog *.v -o Simple_CPU.vvp
vvp Simple_CPU.vvp -fst -sdf-verbose -lxt2
WARNING: Instr_Memory.v:15: $readmemb(Instruction.txt): Not enough words in the file for the requested range [0:64].
LXT2 info: dumpfile Simple_CPU.lxt opened for output.
CONGRATULATION!!
MMMMMMMMMMMMMMMMMMMWXKOOkkOOOXWMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMW0dc:::cc111111cc:::coONMMMMMMMMMMMMM
MMMMMMMMMMMWk:;lkONNNNNNNNNNNNNNNNKko;:dNMMMMMMMMM
MMMMMMMMMO'cONNNNNNNNNNNNNNNNNNNNNNNNKl'xWMMMMMMM
MMMMMMMNc'ONNNNNNNNKXNNNNNNNNNXXNNNNNNNO:;XMMMMMM
MMMMMMW;,KNNNNNNNNO.xNNNNNNNNNl;NNNNNNNNXc'NMMMMM
MMMMWKl.;dXNNOlNNNd.KNNNNNNNNNx.KNNdkNNNxc.:KWMMM
MMXl,cl...ONNd.::::,::::::::::::,::::.cNNX'..cc;cKMM
Wo'oXNNk.,NNNd'WWWWWWWWWWWWWWWWWWWWc:NNNc.dNNNx'cN
O:;.ONX,.:NNNo,Wk:NWWWWWWWWWWWWWooWl;NNNo.'ONK.,;k
MMO.Ox,c.cNNNo,Wl.XWWWWWWWWWWWWW,,Wl,KNNx.:'oK'xMM
MMX.,.:0.lNNNo,Wo'XWWWWWWWWWWWW::Wl'ONNO.do,..OMM
MMMNN.ox.oXNNo,WWWWWWWWWWWWWWWWWWl,kNNX.lk.ONMMM
MMMMO.kl.xKNNo'WWWWWWWNcoocKWWWWWWWl,kONN:;K.dMMMM
MMMMd.K:,xoKNd'WWWWWWWXOkKWWWWWWWc,kxxKo'N;:MMMM
MMMM::N'.','Od.;codxxkkkOOOOkkxdoc;.;d'....Kd.WMMM
MMMN.xNkOXk,.'..kOkxd..looo'.oxkOO'...;xNOxXO.OMMM
MMMx'XNNNNNO....,:oxOO:.dx';kOxo:,..'.kNNNNNN:lMMM
MMW;lNXOKOx..:..:;,,''.......',,;:'.;,.oOKOXNx.NMM
MMO.cc'.ox'.:;.,::::::::::::::::::;.,:'.dx..:l.dMM
MMNOOKl'x,.;:;.;:::::::::::::::::::,':'.d:;XOONMM
MMMMMW';;.Oko,.:::::::::::::::::::''oxO;':.KMMMMM
MMMMMWxoc.,lx,.:::::::::::::::::ccc,.xo;.;odXMMMMM
MMMMMMMMk.Ok..;:::::::::::::::::ccccc:..dK,lMMMMMMMM
MMMMMMMMX'....;:::::::::::::::ccccccc.....OMMMMMMMM
kkmelon@DESKTOP-TEEM7HG:~/ComputerOrganization/LAB4$
```

decoder 的筆記：



Implement detail:

架構圖其實就是照 slide 做的~

Imm_Gen 的 sign-extend 是直接用複製幾個的方式，不太確定有沒有其他方法。

```
/* Write your code HERE */
always @* begin
    casez(opcode)
        7'b0010011, 7'b0000011, 7'b1100111: begin //addi, load
            Imm_Gen_o = {{20{instr_i[31]}}, instr_i[31:20]};
        end
        7'b0100011: begin //store
            Imm_Gen_o = {{20{instr_i[31]}}, instr_i[31:25], instr_i[11:8], instr_i[7]};;
        end
        7'b1100011: begin //Branch
            Imm_Gen_o = {{19{instr_i[31]}}, instr_i[7], instr_i[30:25], instr_i[11:8], 1'b0};
        end
        7'b1101111: begin //jal, jalr
            Imm_Gen_o = {{12{instr_i[31]}}, instr_i[19:12], instr_i[20], instr_i[30:25], instr_i[24:21], 1'b0};
        end
        default: begin
            Imm_Gen_o = 0;
        end
    endcase
end


endmodule
```

ALU_control 是直接沿用上次的~

```
    wire [6-1:0] instr_ALUOp;
    assign instr_ALUOp = {instr, ALUOp};

    // 6'b????00 : I type, S type
    // 6'b????01 : B type
    // 6'b????10 : R type
    always @(*) begin
        casez(instr_ALUOp)
            6'b????00: ALU_Ctrl_o = 4'b0010; // ld sd, add
            6'b????01: ALU_Ctrl_o = 4'b0110; // beq, sub
            6'b000010: ALU_Ctrl_o = 4'b0010; // add
            6'b100010: ALU_Ctrl_o = 4'b0110; // sub
            6'b011110: ALU_Ctrl_o = 4'b0000; // and
            6'b011010: ALU_Ctrl_o = 4'b0001; // or
            6'b001010: ALU_Ctrl_o = 4'b0111; // slt
            6'b110110: ALU_Ctrl_o = 4'b1000; // sra
            6'b000110: ALU_Ctrl_o = 4'b1001; // sll
            6'b010010: ALU_Ctrl_o = 4'b1010; // xor
            default: ALU_Ctrl_o = 4'bxxxx;
        endcase
    end
    endmodule
```

Decoder 是類似這樣給所有 output 值，數字是依照前面ㄉ table

```
/* Write your code HERE */
always @(*) begin
    casez(instr_i)
        7'b0110011: begin //R-type
            RegWrite = 1;
            Branch = 0;
            Jump = 0;
            WriteBack1 = 0;
            WriteBack0 = 0;
            MemRead = 0;
            MemWrite = 0;
            ALUSrcA = 1'bx;
            ALUSrcB = 0;
            ALUOp = 2'b10;
        end
        7'b0010011: begin //addi
```