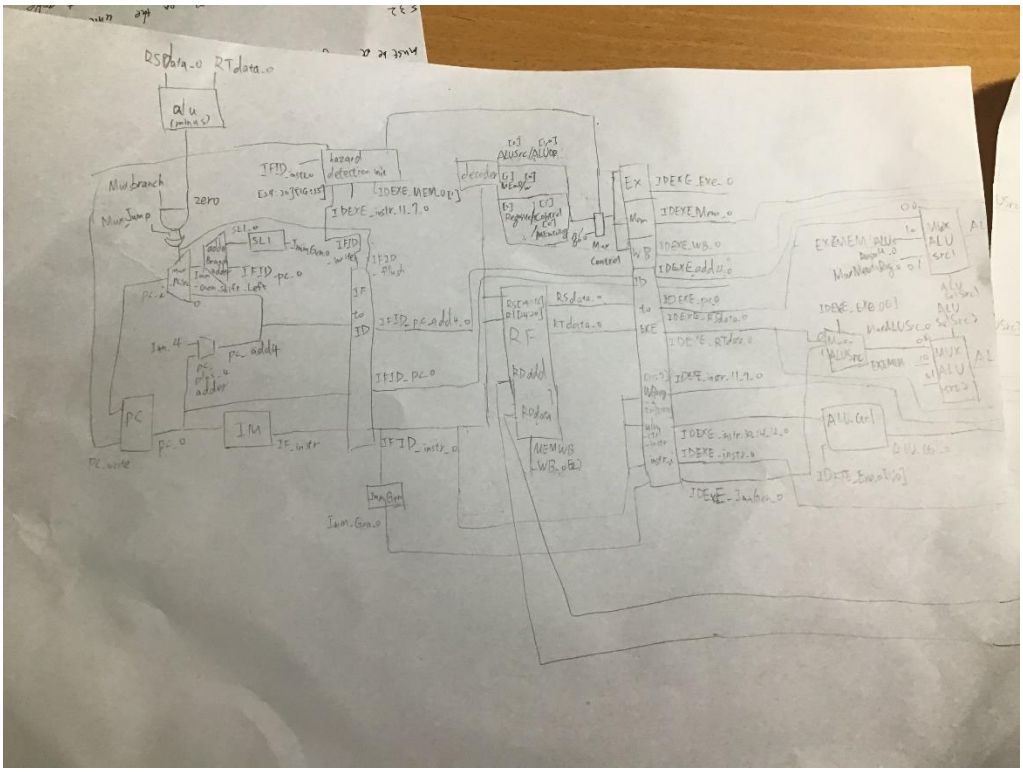
Computer Organization Lab 5 Report

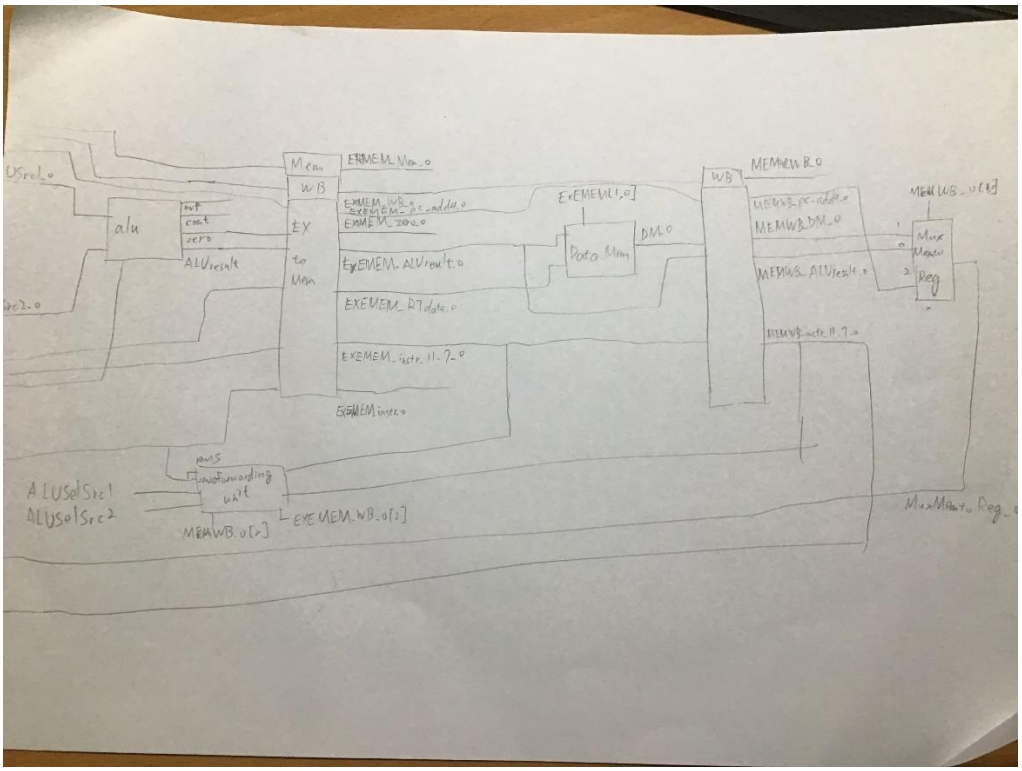
Group10:0711099 林佑檍 0810749 張君實

1. Architecture Diagram

左:



右:



2. Detailed description of the implementation:

助教實作的部分不再贅述

(1) Pipeline_CPU:就照 diagram 接線

(2) alu, ALU_Ctrl, ImmGen, MUX2to1, MUX3to1 就拿 lab4 來用，如果 lab5 有新增指令，就多補。然後 ImmGen 是拿 lab4 來用，已經順便 shift 1 bit 了，所以就算我們有額外寫 Shift_Left_1，最後 Shift_Left_1 還是沒接到線。

(3) IF/EXE/MEM/WB register

IF_register.v - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
module IF_register (clk_i, rst_i,IFID_write,address_i, instr_i,pc_add4_i, address_o, instr_o,flush,pc_add4_o);

        input clk_i ;
        input rst_i;
        input IFID_write;
        input [31:0] address_i;
        input [31:0] instr_i;
        input [31:0] pc_add4_i;
        input flush;
        output reg [31:0] address_o;
        output reg [31:0] instr_o;
        output reg [31:0] pc_add4_o;

always @(posedge clk_i) begin
        if(~rst_i) begin  //reset
                address_o <= 0;
                instr_o <= 0;
                pc_add4_o <= 0;
        end
        else if(flush)begin
                address_o <= 0;
                instr_o <= 0;
                pc_add4_o <= 0;
        end
        else if(IFID_write) begin
                address_o <= address_i;
                instr_o <= instr_i;
                pc_add4_o <= pc_add4_i;
        end
end

endmodule
```

以 IF register 為所有 register 舉例，當~rst_i 時，register output 全歸

零，而其餘狀態就 input 進甚麼，output 出甚麼。而 IF register 有兩個例外，第一個就是 flush，當 flush 發生，就要 output nop(bubble)。而第二個是 IF_ID_write=0 時，就甚麼都不做，讓 output 跟上個 cycle 的 output 一樣。

(4) ProgramCounter

ProgramCounter.v - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
/****************************************************
Student Name:林佑穩_張君實
Student ID: group10_07111099_0810749
****************************************************/

`timescale 1ns/1ps

module ProgramCounter(
    input                               clk_i,
        input                           rst_i,
        input                           PCWrite,
        input      [32-1:0] pc_i,
        output reg [32-1:0] pc_o
        );

//Main function
always @(posedge clk_i) begin
                if(~rst_i)
                        pc_o <= 0;
                else if(PCWrite)
                        pc_o <= pc_i;
                else if(~PCWrite)
                        pc_o <= pc_i-4;
        end

endmodule
```

當~rst_i 時，pc_o=0。如果 PCWrite，就是將 input 的東西 output 出去。如果是 other，那就把上個 cycle 的 output 再 output 一遍。

(5) ForwardingUnit

```
module ForwardingUnit (EXE_instr19_15, EXE_instr24_20, MEM_instr11_7, MEM_WBControl, WB_instr11_7, WB_Control, src1_sel_o, src2_sel_o);

        input [5-1:0] EXE_instr19_15,   //RS1
                      EXE_instr24_20,   //RS2
                      MEM_instr11_7,    //MEM_RD
                      WB_instr11_7;     //WB_RD
        input MEM_WBControl,
              WB_Control;
        output wire [2-1:0] src1_sel_o,
                            src2_sel_o;

reg [2-1:0] SLT1;
reg [2-1:0] SLT2;
assign src1_sel_o = SLT1;
assign src2_sel_o = SLT2;

always @(*) begin
        if(MEM_WBControl == 1'b1 && (MEM_instr11_7!=0) && EXE_instr19_15 == MEM_instr11_7) begin
                SLT1 <= 2'b10;
        end
        else if(WB_Control == 1'b1 && (WB_instr11_7!=0) && EXE_instr19_15 == WB_instr11_7 && (!(MEM_WBControl == 1'b1 && (MEM_instr11_7!=0) && EXE_instr19_15 == MEM_instr11_7))) begin
                SLT1 <= 2'b01;
        end
        else begin
                SLT1 <= 2'b00;
        end

        if(MEM_WBControl == 1'b1 && (MEM_instr11_7!=0) && EXE_instr24_20 == MEM_instr11_7) begin
                SLT2 <= 2'b10;
        end
        else if(WB_Control == 1'b1 && (WB_instr11_7!=0) && EXE_instr24_20 == WB_instr11_7 && (!(MEM_WBControl == 1'b1 && (MEM_instr11_7!=0) && EXE_instr24_20 == MEM_instr11_7))) begin
                SLT2 <= 2'b01;
        end
        else begin
                SLT2 <= 2'b00;
        end
end

endmodule
```

照著講義演算法，如果 memRd/wbRd 和 exeRs1/exeRs2 相符，且 memread/wbread 為 1，memRd/wbRd 不為 0，那就要 output 不同的

SLT 來讓 alu 前的 mux input 進去正確 cycle 的數值。如果 memRd 和 wbRd 和 exeRs1/exeRs2 同時相符，那就 output 對應到 mem input 的 SLT。

(6) Hazard_detection

```
module Hazard_detection(
input [4:0] IFID_regRs,
input [4:0] IFID_regRt,
input [4:0] IDEXE_regRd,
input IDEXE_memRead,
output reg PC_write,
output reg IFID_write,
output reg control_output_select
);

always@(*)begin
    if((IDEXE_memRead == 1'b1) && ((IDEXE_regRd == IFID_regRs)||(IDEXE_regRd == IFID_regRt)))begin
        PC_write <= 1'b0;
        IFID_write <= 1'b0;
        control_output_select <= 1'b0;
    end

    else begin
        PC_write <= 1'b1;
        IFID_write <= 1'b1;
        control_output_select <= 1'b1;
    end
end

endmodule
```

照講義演算法實作 load-use hazard，如果 EXE_memRead=1 且 (EXErd=IDrs 或 EXErd=IDrt)，就會觸發 hazard detection，讓 PC_write, IFID_write 和 contorl_output_select 為 0，也就是在 exe 產生 nop，exe 以前的 layer 就延遲一個 cycle，exe 以後的 layer 就正常運作。而其他情形就正常運作。

(7) branch hazard

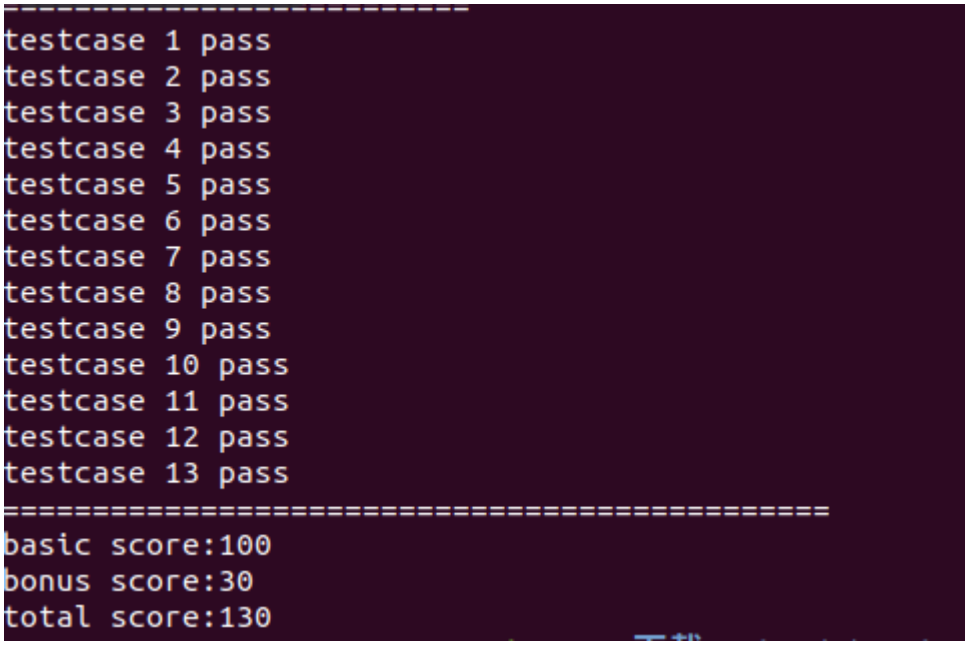
可以參考 diagram，我們在 ID/EXE 間額外有裝一個 alu 計算相減後的 zero，並讓(zero&&branch)||Jump 得出 PCSrc，如果 PCSrc 為 1，那就 flush IFID_register 並讓 ProgramCounter 跳到 branch 後的位置。

(8) branch adder 與 MuxMemToReg

參考 lab4 的做法，而 MuxMemToReg 是 3 元 mux，control 分別為 {jump,MemToReg}。

3. Result:

透過 lab5TestScript.sh，可以得出以下滿分的結果。



4. Comment:

這次的 lab 沒有完整的 diagram，再加上我不太會 debug，所以寫起來滿痛苦的，不過最後我們這組還是生出來了，快累死了，希望下次助教能放人一馬。