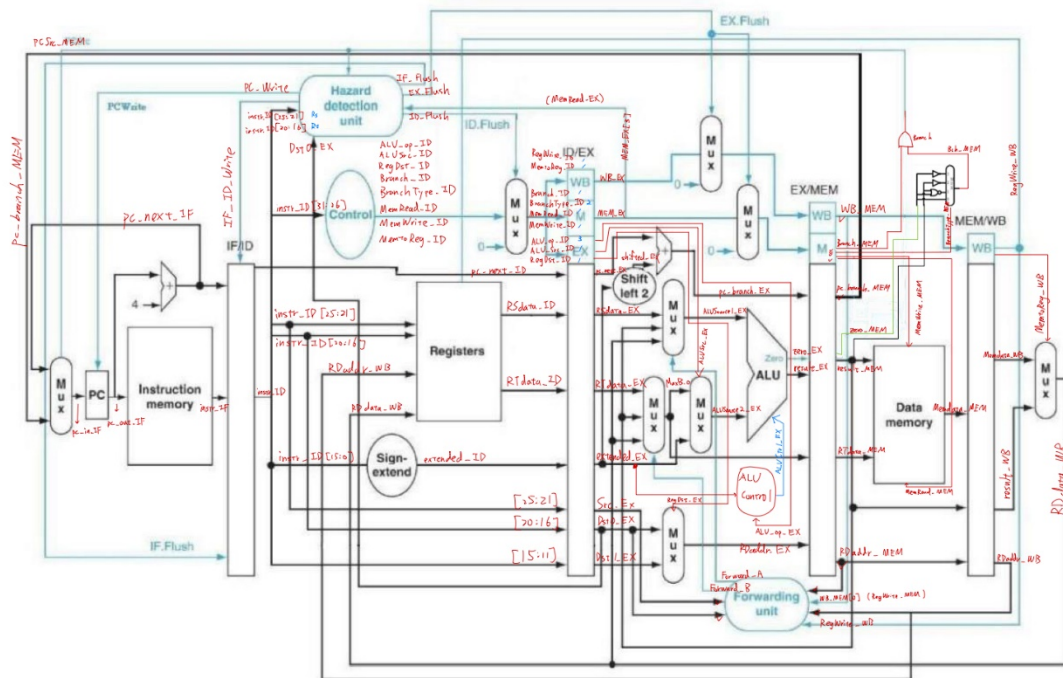


Computer Organization Lab5

Name: 陳子祈

ID: 0819823

Architecture diagrams:



Hardware module analysis:

1. forwarding:

input [4:0] EX_Rs;

input [4:0] EX_Rt;

input [4:0] MEM_Rd;

input MEM_RegWrite;

input [4:0] WB_Rd;

input WB_RegWrite;

output [1:0] Forward_A;

output [1:0] Forward_B;

description:

(1) 檢查 EX_Rs、EX_Rt 是否為 MEM_RD 及 MEM_RegWrite 是否是 1 及 MEM_RD 是否為

非 0，如果以上條件都成立則使用 Forwarding，以此來解決 MEM 的 data hazard。

- (2) 若 MEM_Rd == EX_Rs 或 EX_Rt 不成立，接著再檢查 EX_Rs、EX_Rt 是否為 WB_Rd 且 WB_RegWrite 是否為 1 及 WB_Rd 是否為非 0，若上述條件都成立則繼續使用 Forwarding，以此來解決 WB 的 data hazard。在發現 WB 的 data hazard 並做 Forwarding 之前，要先排除 MEM 的 data hazard 的發生才能使用 Forwarding，避免 double data hazard 的發生。

2. HazardDetection:

```
input EX_MemRead;
input [4:0]EX_Rt;
input [4:0] ID_Rs;
input [4:0] ID_Rt;
input PCSrc;
output PC_Write;
output IF_ID_Write;
output IF_Flush;
output ID_Flush;
output EX_Flush;
```

description:

HazardDetection主要是在偵測Load use data hazard或Branch是否發生，並作相對應的動作。

- (1) 如果EX_MemRead ==1 且 EX_Rt==ID_Rs或ID_Rt 則代表Load use data hazard發生，這時要stall one block，將PC及IF/ID_Reg保存起來，就是設定result_PC_Write=0且result_IF_ID_Write=0，除此之外，還需要製造一個bubble，也就是設定result_ID_Flush=1。
- (2) 當PCSrc為1代表Branch發生，此時要防止前面幾個不需要執行的指令更新暫存器或記憶體，將IF、ID、EX正在執行的指令都Flush，也就是把IF/ID、ID/EX、EX/MEM的Pipeline Register全部變成0，因此輸出IF_Flush、ID_Flush、EX_Flush皆為1。

其他 module 之前都已經有詳細說明，在此 LAB 就不贅述。

Finished part:

請助教在測試 testbench.v 的時候可以將 MAX_COUNT 調成 64，如下：

```

42 for(i=0; i<16; i=i+1)
43 begin
44     cpu.IM.instruction_file[i] = 32'b0;
45 end
46
47 $readmemb("CO_P5_test_1.txt", cpu.IM.instruction_file); //Read instruction from "CO_P4_test_1.txt"
48
49 // data memory
50 for(i=0; i<128; i=i+1)
51 begin
52     cpu.DM.Mem[i] = 8'b0;
53 end
54
55 #((CYCLE_TIME) RST = 1;
56 #((CYCLE_TIME*64) $stop;
57 //((CYCLE_TIME*20) $fclose(handle); $stop;
58 end
59
60 //Print result to "CO_P4_Result.dat"
61 always@(posedge CLK) begin
62     count = count + 1;

```

For CO_P5_test_1.txt:

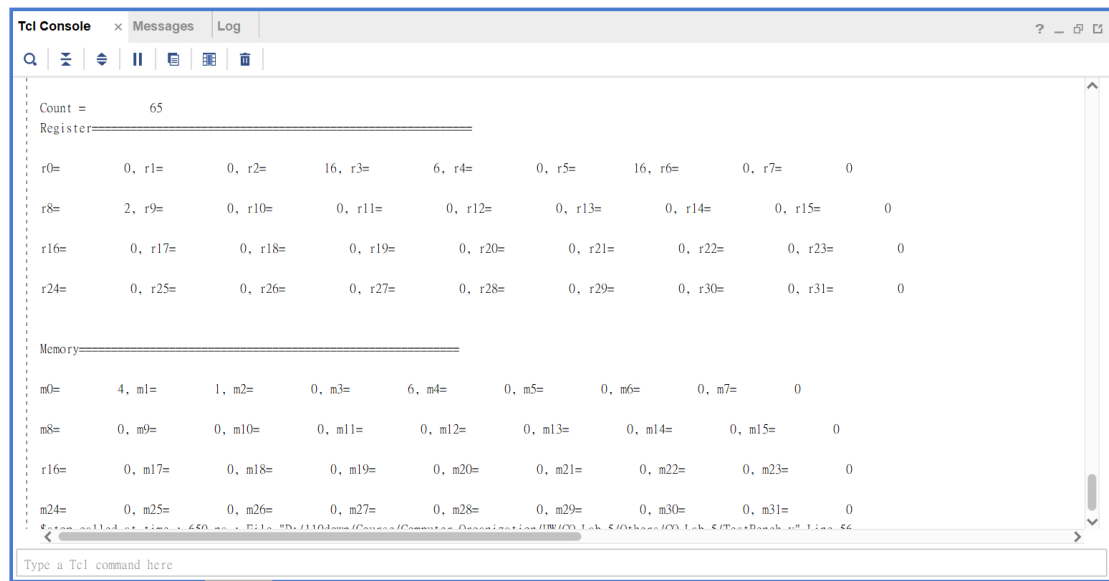
```

Tcl Console x Messages Log
Count = 71
Register=====
r0= 0, r1= 16, r2= 256, r3= 8, r4= 16, r5= 8, r6= 24, r7= 26
r8= 8, r9= 1, r10= 0, r11= 0, r12= 0, r13= 0, r14= 0, r15= 0
r16= 0, r17= 0, r18= 0, r19= 0, r20= 0, r21= 0, r22= 0, r23= 0
r24= 0, r25= 0, r26= 0, r27= 0, r28= 0, r29= 0, r30= 0, r31= 0

Memory=====
m0= 0, m1= 16, m2= 0, m3= 0, m4= 0, m5= 0, m6= 0, m7= 0
m8= 0, m9= 0, m10= 0, m11= 0, m12= 0, m13= 0, m14= 0, m15= 0
r16= 0, m17= 0, m18= 0, m19= 0, m20= 0, m21= 0, m22= 0, m23= 0
m24= 0, m25= 0, m26= 0, m27= 0, m28= 0, m29= 0, m30= 0, m31= 0
$stop called at time : 710 ns : File "D:/110down/Course/Computer_Organization/HW/CO_Lab_5/Others/CO_Lab_5/TestBench.v" Line 56
elapsed sim Time (s) : 00:00:01, elapsed = 00:00:00, Memory (MB) : peak = 1268.780, avail = 0.000

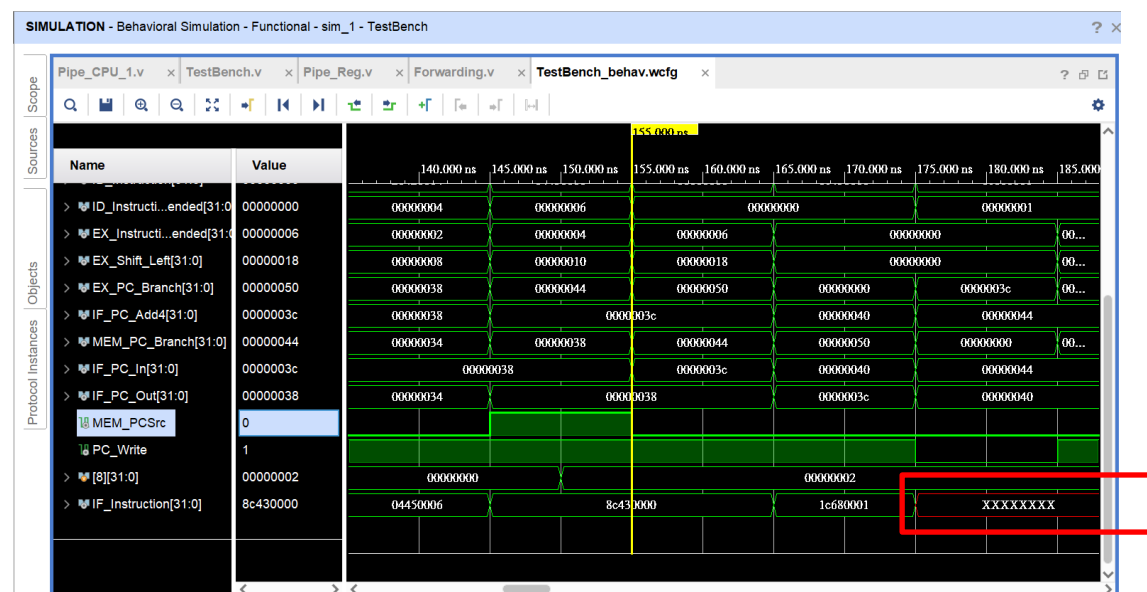
```

For CO_P5_test_2.txt:

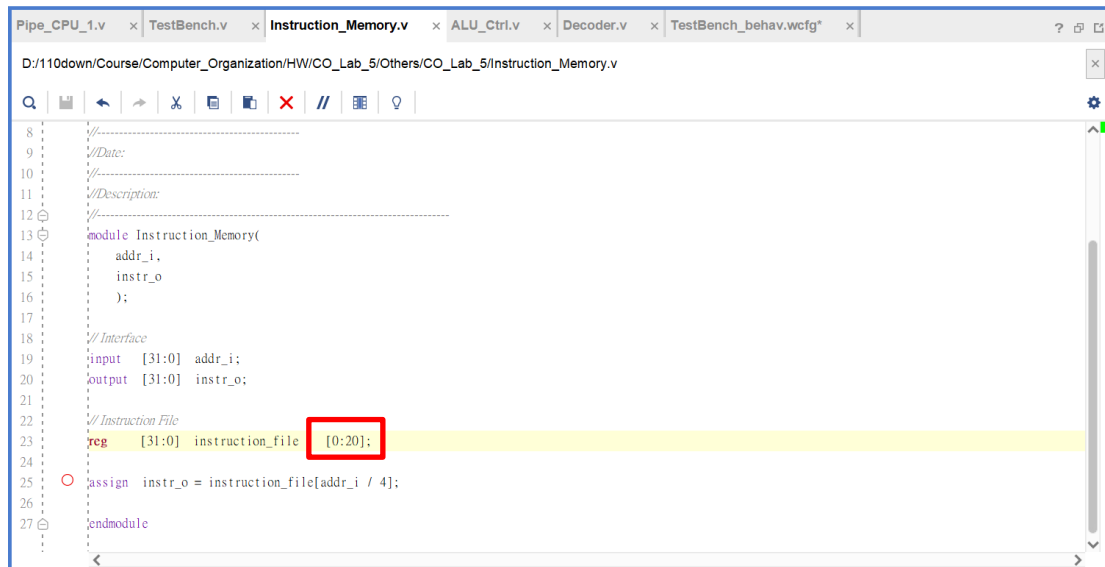


Problems you met and solutions:

Q1: 發現 IF_Instruction 會讀不到值。



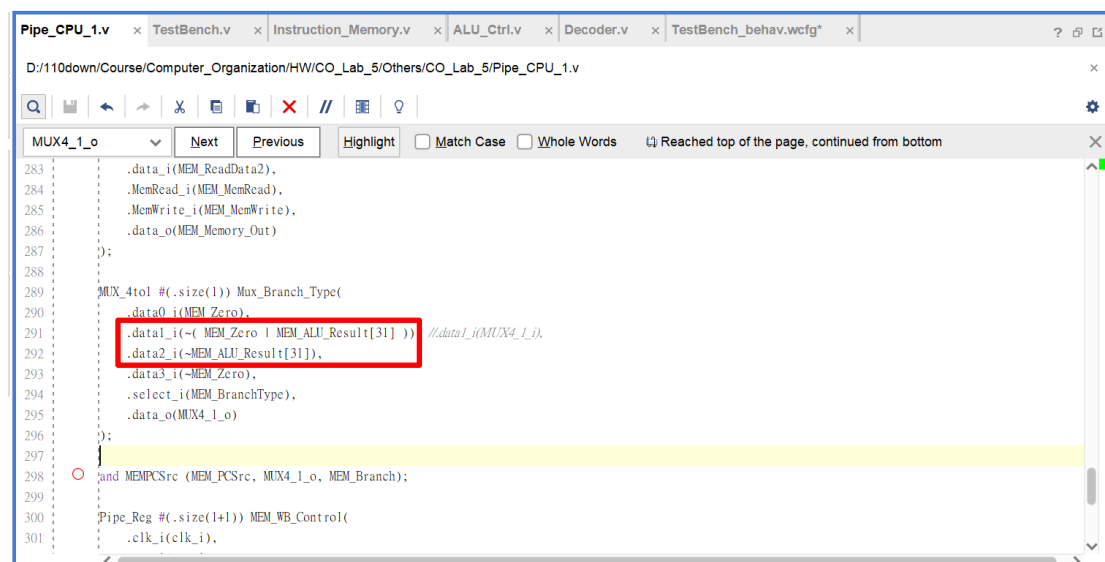
A1: 因為我原本 instruction_file 設為 15，但實際上指令有 21 個，所以後面會有一些指令讀不到



```
8 //-----
9 //Date:
10 //-----
11 //Description:
12 //-----
13 module Instruction_Memory(
14     addr_i,
15     instr_o
16 );
17
18 // Interface
19 input  [31:0] addr_i;
20 output [31:0] instr_o;
21
22 // Instruction File
23 reg [31:0] instruction_file [0:20];
24
25 assign instr_o = instruction_file[addr_i / 4];
26
27 endmodule
```

Q2: bgt、bne 寫錯，找很久才發現有錯。

A2: 原本以為 branch 指令跳到錯誤的指令位置，後來發現是我自己 bgt、bne 寫錯，改正確就好了。



```
283 .data_i(MEM_ReadData2),
284 .MemRead_i(MEM_MemRead),
285 .MemWrite_i(MEM_MemWrite),
286 .data_o(MEM_Memory_Out)
287 );
288
289 MUX4_t0l #(Mux_Branch_Type(
290     .data0_i(MEM_Zero),
291     .data1_i(~MEM_Zero | MEM_ALU_Result[31]) //data1_i(MUX4_1_i),
292     .data2_i(~MEM_ALU_Result[31]),
293     .data3_i(~MEM_Zero),
294     .select_i(MEM_BranchType),
295     .data_o(MUX4_1_o)
296 ));
297
298 and MEMPCSrc (MEM_PCSrc, MUX4_1_o, MEM_Branch);
299
300 Pipe_Reg #(Mux_WB_Control(
301     .clk_i(clk_i),
```

Summary:

這次的 LAB 比之前難很多，雖然只有加 Forward 及 Hazard detection 兩個模組，可是要注意很多細節，尤其是判斷 hazard 的部分。經過學長提醒，我才知道我後面很多指令讀不到原來是因為 Instruction_file 設太少。另外，我還以為遇到 Branch 的問題，結果原來是自己寫錯 bgt 及 bge。這次學到很多，希望暑假再好好精進實作的能力，我覺得自己很長粗心，找錯要花很多時間，我花了整整三天才搞清楚問題在哪裡，希望下次可以好好加油。