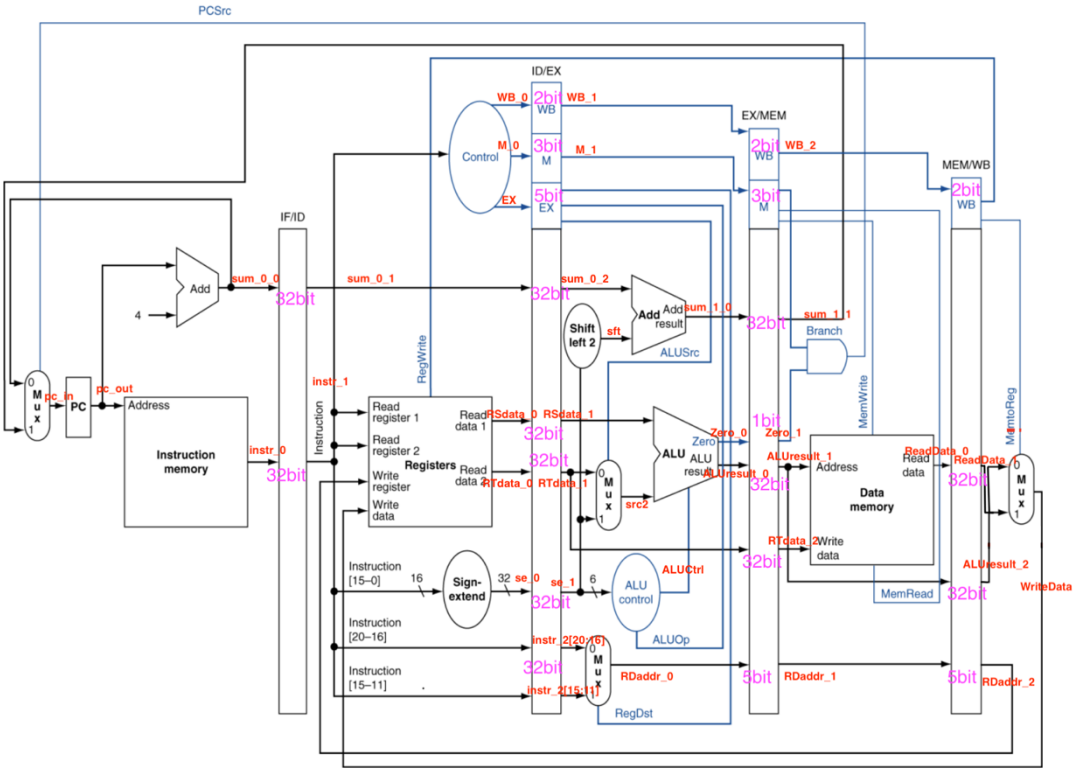
Computer Organization Lab4

ID: 110550029 Name: 陳芷萱

//以 iverilog 編譯

Architecture diagrams



Hardware module analysis

本次作業的設計以 Lab3 為基礎，在以下模組做更動：

- Decoder.v

  移除了 J-type 指令之 Opcode 判讀，以及 Link_o、jump_o、all_zeros_o 三個輸出。

- Alu_Ctrl.v

  若根據輸入的 ALUOp 判斷指令為 R-type，新增 XOR 以及 mult 兩種指令的判斷，並分別輸出 0011 以及 1111 的控制訊號 ALUCtrl。

```
//Parameter
parameter R=3'b011;

//Select exact operation
always @(funct_i, ALUOp_i) begin
        if(ALUOp_i==R) begin
                case(funct_i)
                        24: ALUCtrl_o<=4'b1111; //mult
                        32: ALUCtrl_o<=4'b0010; //add
                        34: ALUCtrl_o<=4'b0110; //sub
                        36: ALUCtrl_o<=4'b0000; //AND
                        37: ALUCtrl_o<=4'b0001; //OR
                        38: ALUCtrl_o<=4'b0011; //XOR
                        42: ALUCtrl_o<=4'b0111; //slt
                endcase
        end
        else begin
                ALUCtrl_o[2:0]<=ALUOp_i;
                ALUCtrl_o[3]=1'b0;
        end
end
```

- Alu.v

  根據 ALU_Ctrl 輸出的控制訊號 ctrl_i，新增了 XOR 以及 mult 兩指令的判斷以及計算功能。

```
//Main function
assign zero_o=(result_o==0);
always @(ctrl_i, src1_i, src2_i) begin
        case(ctrl_i)
                0: result_o <= src1_i&src2_i; //AND
                1: result_o <= src1_i|src2_i; //OR
                2: result_o <= src1_i+src2_i; //add
                3: result_o <= src1_i^src2_i; //XOR
                6: result_o <= src1_i-src2_i; //sub
                7: result_o <= src1_i<src2_i ? 1:0; //slt
                12: result_o <= ~(src1_i|src2_i); //NOR
                15: result_o <= src1_i*src2_i; //mult
                default: result_o <= 0;
        endcase
end
```

- Pipelined_CPU.v

  根據 Architecture Diagram 之設計將模組連接，其中 MUX 之輸入皆為 32 bit，Pipe_Reg 之輸入則根據欲傳至下一 state 的資料長度總和決定。

  IF/ID：32+32=64 bit

  ID/EX：2+3+5+32+32+32+32+32=170 bit

  EX/MEM：2+3+32+1+32+32+5=107 bit

  MEM/WB：2+32+32+5=71 bit

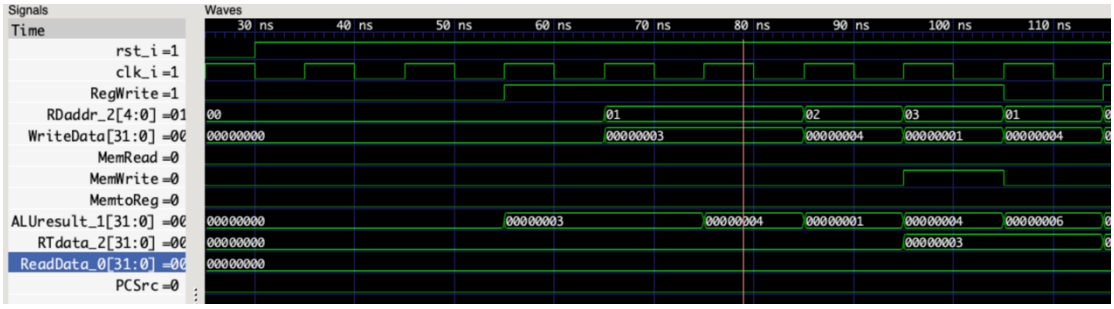
  輸出與輸入方式則是根據 Architecture Diagram，依序將訊號放入大括號內輸入、取出。

```
Pipe_Reg #(.size(107)) EX_MEM( // Modify N, which is the total length of input/output
        .clk_i(clk_i),
        .rst_i(rst_i),
        .data_i({WB_1, M_1, sum_1_0, Zero_0, ALUresult_0, RTdata_1, RDaddr_0}),
        .data_o({WB_2, Branch, MemRead, MemWrite, sum_1_1, Zero_1, ALUresult_1, RTdata_2, RDaddr_1})
);
```

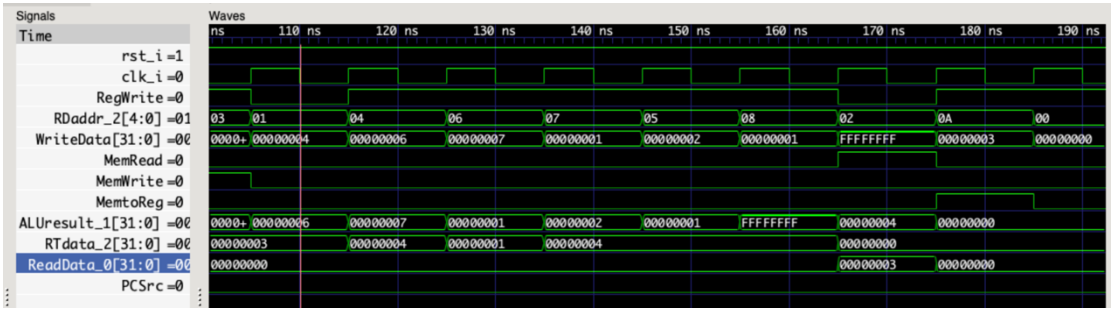
Simulation results

- Testcase1

根據輸出之波形圖，在 reset 訊號為 1 後，
第 5 個 clock cycle 時，3 被寫入 r1；
第 6 個 clock cycle 時，4 被寫入 r2；
第 7 個 clock cycle 時，1 被寫入 r3，且 3（=r1）被寫入 m[1]；
第 8 個 clock cycle 時，未做暫存器或記憶體讀寫；



第 9 個 clock cycle 時，6（=r1+r1）被寫入 r4；
第 10 個 clock cycle 時，7（=r1 | r2）被寫入 r6；
第 11 個 clock cycle 時，1（=r1 & r3）被寫入 r7；
第 12 個 clock cycle 時，2（=r4 – r2）被寫入 r5；
第 13 個 clock cycle 時，1（=r1<r2）被寫入 r8，且 PCSrc 為 0，不進行 branch（r1 與 r2 值不相等）；
第 14 個 clock cycle 時，記憶體讀出 3（=m[1]），不進行暫存器寫入；
第 15 個 clock cycle 時，3（第 14 個 clock cycle 從記憶體讀出）被寫入 r10。

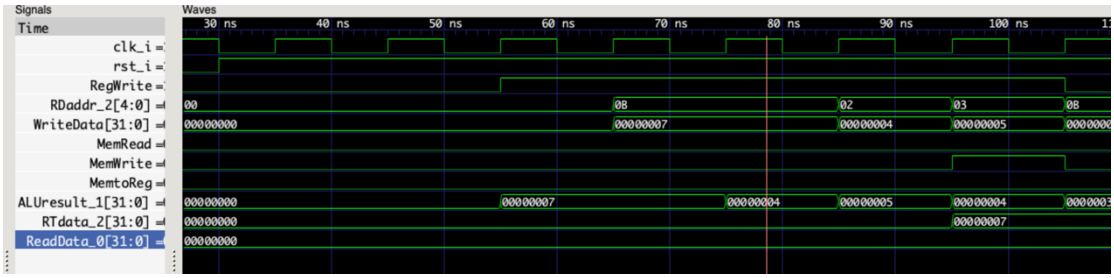


最終結果亦與解答相符：

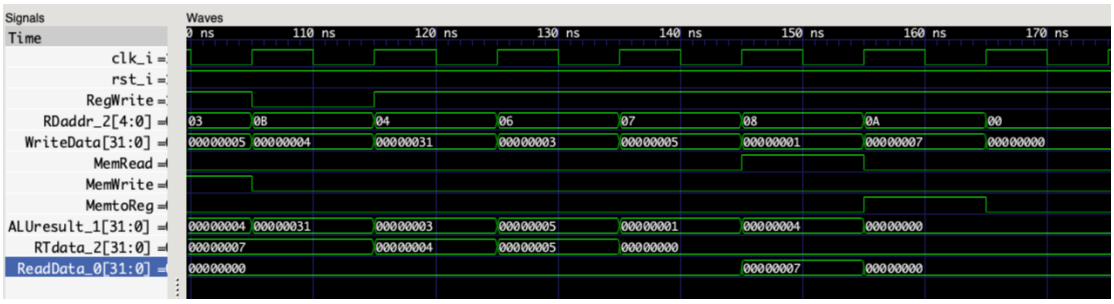
```
- Register File -
 r0 =     0      r1 =     3      r2 =     4      r3 =     1
 r4 =     6      r5 =     2      r6 =     7      r7 =     1
 r8 =     1      r9 =     0     r10 =     3     r11 =     0
r12 =     0     r13 =     0     r14 =     0     r15 =     0
r16 =     0     r17 =     0     r18 =     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 Data -
 m0 =     0      m1 =     3      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
m16 =     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
```

- Testcase2

根據輸出之波形圖，在 reset 訊號為 1 後，
第 5 個 clock cycle 時，7 被寫入 r11；
第 6 個 clock cycle 時，4 被寫入 r2；
第 7 個 clock cycle 時，5 被寫入 r3，7（=r11）被寫入 m[1]；



第 8 個 clock cycle 時，未做暫存器或記憶體讀寫；
第 9 個 clock cycle 時，49（=r11*r11）被寫入 r4；
第 10 個 clock cycle 時，3（=r11^r2）被寫入 r6；
第 11 個 clock cycle 時，5（=r11 & r3）被寫入 r7；
第 12 個 clock cycle 時，1（=r11<10）被寫入 r8，並從記憶體讀出 7（=m[1]）；
第 13 個 clock cycle 時，7（第 12 個 clock cycle 從記憶體讀出）被寫入 r10。



最終結果亦與解答相符：

```
- Register File -
 r0 =     0      r1 =     0      r2 =     4      r3 =     5
 r4 =    49      r5 =     0      r6 =     3      r7 =     5
 r8 =     1      r9 =     0     r10 =     7     r11 =     7
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 Data -
 m0 =     0      m1 =     7      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
m16 =     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
```

- Testcase3
  - **指令重排序：**

    I2 需在 I1 後至少 3 個 clock cycle 才可執行（此時 r1 的值才會更新成功），因此將 I2 挪至 I4 後方；

    I6 需在 I5 後至少 3 個 clock cycle 才可執行（此時 r4 的值才會更新成功），因此將 I6 挪至 I8 後方；

    I9 需在 I8 後至少 3 個 clock cycle 才可執行（此時 r7 的值才會更新成功），因此將 I9 挪至 I10 後方。

    最終指令順序更改為：I1 -> I3 -> I4 -> I2 -> I5 -> I7 -> I8 -> I6 -> I10 -> I9

```
00100000000000010000000000010000
00100000000000011000000000001000
10101100000000010000000000000100
00100000001000100000000000000100
10001100000001000000000000000100
00000000011000010011000000100000
00100000001001110000000000001010
00000000100000110010100000100010
00100000000010010000000001100100
00000000111000110100000000100100
```

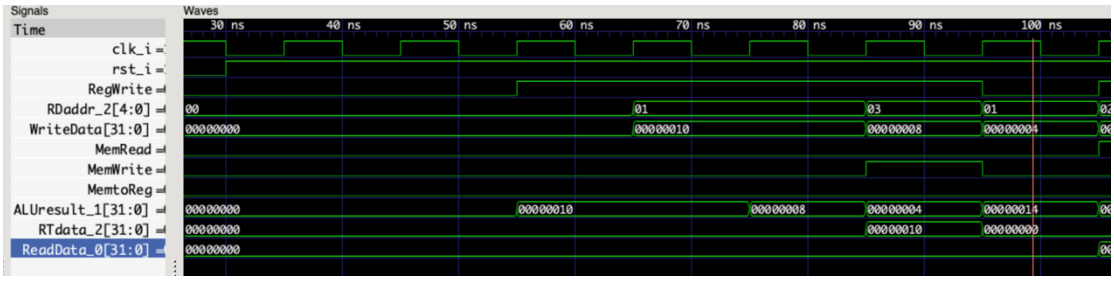
  - **結果分析：**

    根據輸出之波形圖，在 reset 訊號為 1 後，

    第 5 個 clock cycle 時，16 被寫入 r1（I1）；

    第 6 個 clock cycle 時，8 被寫入 r3（I3），16（=r1）被寫入 m[1]（I4）；

    第 7 個 clock cycle 時，未做暫存器或記憶體讀寫；



    第 8 個 clock cycle 時，20 被寫入 r2（=r1+4）（I2），並從記憶體讀出 16（=m[1]）（I5）；

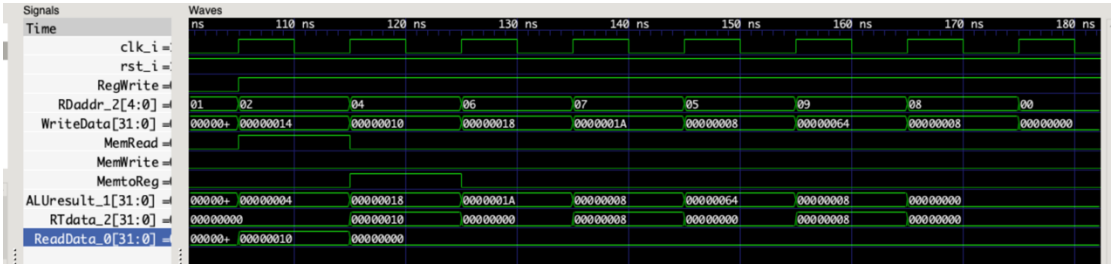
    第 9 個 clock cycle 時，16（第 8 個 clock cycle 從記憶體讀出）被寫入 r4（I5）；

    第 10 個 clock cycle 時，24（=r1+r3）被寫入 r6（I7）；

    第 11 個 clock cycle 時，26（=r1+10）被寫入 r7（I8）；

    第 12 個 clock cycle 時，100 被寫入 r9（I10）；

    第 13 個 clock cycle 時，8（=r7 & r3）被寫入 r8（I9）。



最終結果亦與解答相符：

```
- Register File -
 r0 =     0      r1 =    16      r2 =    20      r3 =     8
 r4 =    16      r5 =     8      r6 =    24      r7 =    26
 r8 =     8      r9 =   100     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 Data -
 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
m16 =     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
```

Problems you met and solutions

此次作業在了解 Pipelined CPU 的架構並設計完電路圖後，稍微根據題目要求修改模組、連接模組即可完成，並未遇到太大的問題。唯一遇到的小問題是將 Control 輸出的訊號輸入 M 存放時，將頭尾順序弄反，因此花了一點時間 trace 波形圖 debug。

Summary

原本認為 Pipelined CPU 在實作上會困難許多，但經過這次作業後發現基本上僅是多了一個放入暫存器再取出的動作，並沒有想像中的困難，在這次作業後也更了解了 Pipelined CPU 的執行流程。