Computer Organization

1. The input fields of each pipeline register:

```
Register #(.size(64)) IFID(
    .clk_i(clk_i),
    .rst_n(rst_n),
    .data_i({PC_add1,instr}),
    .data_o(IFID_o)
);
```

```
Register #(.size(148)) IDEX(
    .clk_i(clk_i),
    .rst_n(rst_n),
    .data_i({
        RegWrite,        //WB    [147]
        MemtoReg[0],     //WB    [146]
        Branch,          //M     [145]
        MemRead,         //M     [144]
        MemWrite,        //M     [143]
        RegDst[0],       //EX    [142]
        ALUOP[2:0],      //EX    [141:139]
        ALUSrc,          //EX    [138]
        IFID_o[64-1:32],//32    [137:106]
        ReadData1,       //32    [105:74]
        ReadData2,       //32    [73:42]
        signextend,      //32    [41:10]
        IFID_o[20:16],   //5     [9:5]
        IFID_o[15:11]    //5     [4:0]
        }),
    .data_o(IDEX_o)
);
```

```
Register #(.size(107)) EXMEM(
    .clk_i(clk_i),
    .rst_n(rst_n),
    .data_i({
        IDEX_o[147:146],//WB-2   [106:105]
        IDEX_o[145:143],//M-3    [104:102]
        branch_address[31:0],//  [101:70]
        zero,//                  [69]
        alu_res[31:0],//         [68:37]
        IDEX_o[73:42],//         [36:5]
        write_address[4:0]//     [4:0]
        }),
    .data_o(EXMEM_o)
);
```

```
Register #(.size(71)) MEMWB(
    .clk_i(clk_i),
    .rst_n(rst_n),
    .data_i({
        EXMEM_o[106:105],//WB-2  [106:105]
        read_data,       //[68:37]
        EXMEM_o[68:37],  //[37:5]
        EXMEM_o[4:0]     //[4:0]
        }),
    .data_o(MEMWB_o)
);
```

2. Compared with lab4, the extra modules:

加上了 register.V 檔來建構 各 stage 之間的 cpu。

從原本的 single cycle cpu 加上了 4 個 register 來儲存每個 pipeline 的 data 和 signal 變成 Pipeline cpu。

3. Explain your control signals in sixth cycle (both test patterns C0_P5_test_data1 and C0_P5_test_data2 are needed):

Picture:

CO_P5_test_data1:

Addi	IF	ID	EX	MEM	WB	
Addi		IF	ID	EX	MEM	WB
Addi			IF	ID	EX	MEM
And				IF	ID	EX
Or					IF	ID
slt						IF

→

EX			MEM			WB	
RegDst	ALUOP	ALUSrc	Branch	MemRead	MemWrite	RegWrite	MemtoReg
1	010	0	0	0	0	1	0

CO_P5_test_data2:

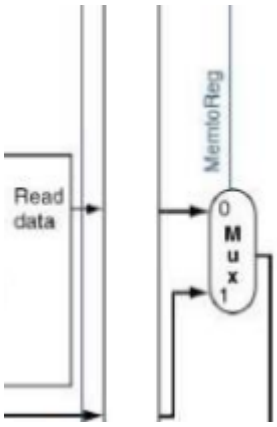
Addi	IF	ID	EX	MEM	WB	
Addi		IF	ID	EX	MEM	WB
Addi			IF	ID	EX	MEM
Addi				IF	ID	EX
Sw					IF	ID
Sw						IF
sub						

→

EX			MEM			WB	
RegDst	ALUOP	ALUSrc	Branch	MemRead	MemWrite	RegWrite	MemtoReg
0	011	1	0	0	0	1	0

4. Problems you met and solutions:

當我寫完之後跑 testbench 發現我的 resister 都是 X 我就把每個 module 都每條線都印出來，從第一個印到最後一個才發現圖是錯的這裡 1 0 寫反了改了之後就對了



原本：

```
Mux2to1 #(.size(32)) write_data(
    .data0_i(MEMWB_o[68:37]),
    .data1_i(MEMWB_o[36:5]),
    .select_i(MEMWB_o[69]),
    .data_o(write_data_o)
);
```

後來：

```
Mux2to1 #(.size(32)) write_data(
    .data0_i(MEMWB_o[36:5]),
    .data1_i(MEMWB_o[68:37]),
    .select_i(MEMWB_o[69]),
    .data_o(write_data_o)
);
```

5. Summary:

原本上課時只是知道 pipeline cpu 運作的原理，這次的作業讓我們從之前寫的 single cycle cpu 加上 register 轉化成 pipeline cpu，讓我們對 pipeline cpu 的運作更理解。