Computer Organization

1. The input fields of each pipeline register:

IF/ID: {PC_add1, instr} (64 bits)

ID/EX: {RegWrite, MemtoReg[0], Branch, MemRead, MemWrite, RegDst[0], BranchType, ALUOP[2:0], ALUSrc, IFID_o[64-1:32], ReadData1, ReadData2, signextend, IFID_o[20:0]} (160 bits)

EX/MEM: {IDEX_o[191:187], PC_add2, zero, ALUResult, IDEX_o[116:85], Mux_Write_o} (107 bits)

MEM/WB: {EXMEM_o[106:105], DM_ReadData, EXMEM_o[68:37], EXMEM_o[4:0]} (71 bits)

2. Compared with lab4, the extra modules:

除了沿用前幾個 Lab 的 1-bit ALU 以及 Full Adder 等 module 以外，我將 Single Cycle CPU 改成 Pipeline CPU。

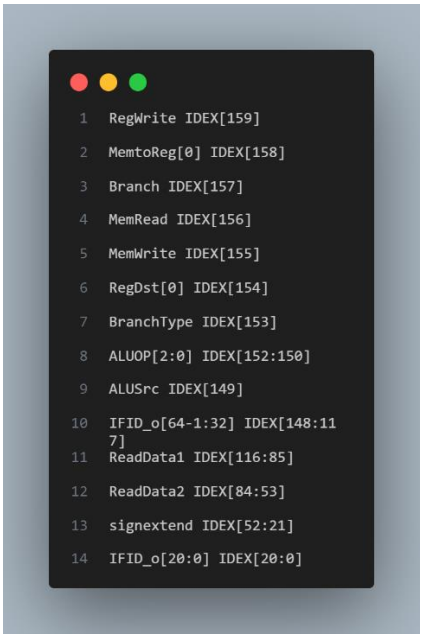
此外，我還有新增了 Pipeline_Reg.v，也就是圖中的 IF/ID、ID/EX、EX/MEM、MEM/WB 這四個 register。

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

C0_P5_test_data1	C0_P5_test_data2
MemtoReg: 0 RegWrite: 1 MemRead: 0 MemWrite: 0 RegDst: 1 PCSrc: 0 AluOP: 010 ALUSrc: 0	RegWrite: 1 MemtoReg: 0 RegDst: 0 PCSrc: 0 AluOP: 011 ALUSrc: 1 MemRead: 0 MemWrite: 0

4. Problems you met and solutions:

我遇到的問題是 4 個 Pipeline Register 的 index 常常接錯，特別是 ID/EX 的部分，因為有 160 bits 又有很多 control signals，最後是我開記事本一個一個慢慢打出來(如下圖)才找到錯誤並修正 index。



5. Summary:

在這次的作業當中，我學到了如何利用 Register 將原本的 Single Cycle CPU 切成五個階段，將其中的 Data 以及 Signals 暫存起來，並且利用 Pipeline 讓原本的 Instruction 能夠多工處理，將原本所需要的時間壓到更少。