# **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 CO\_P5\_test\_data1 and CO\_P5\_test\_data2 are needed):

CO_P5_test_data1	CO_P5_test_data2
MemtoReg: 0	RegWrite: 1
RegWrite: 1	MemtoReg: 0
MemRead: 0	RegDst: 0
MemWrite: 0	PCSrc: 0
RegDst: 1	AluOP: 011
PCSrc: 0	ALUSrc: 1
AluOP: 010	MemRead: 0
ALUSrc: 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。

```
1 RegWrite IDEX[159]
2 MemtoReg[0] IDEX[158]
3 Branch IDEX[157]
4 MemRead IDEX[156]
5 MemWrite IDEX[155]
6 RegDst[0] IDEX[154]
7 BranchType IDEX[153]
8 ALUOP[2:0] IDEX[152:150]
9 ALUSrc IDEX[149]
10 IFID_o[64-1:32] IDEX[148:11 7]
11 ReadData1 IDEX[116:85]
12 ReadData2 IDEX[84:53]
13 signextend IDEX[52:21]
14 IFID_o[20:0] IDEX[20:0]
```

### 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。