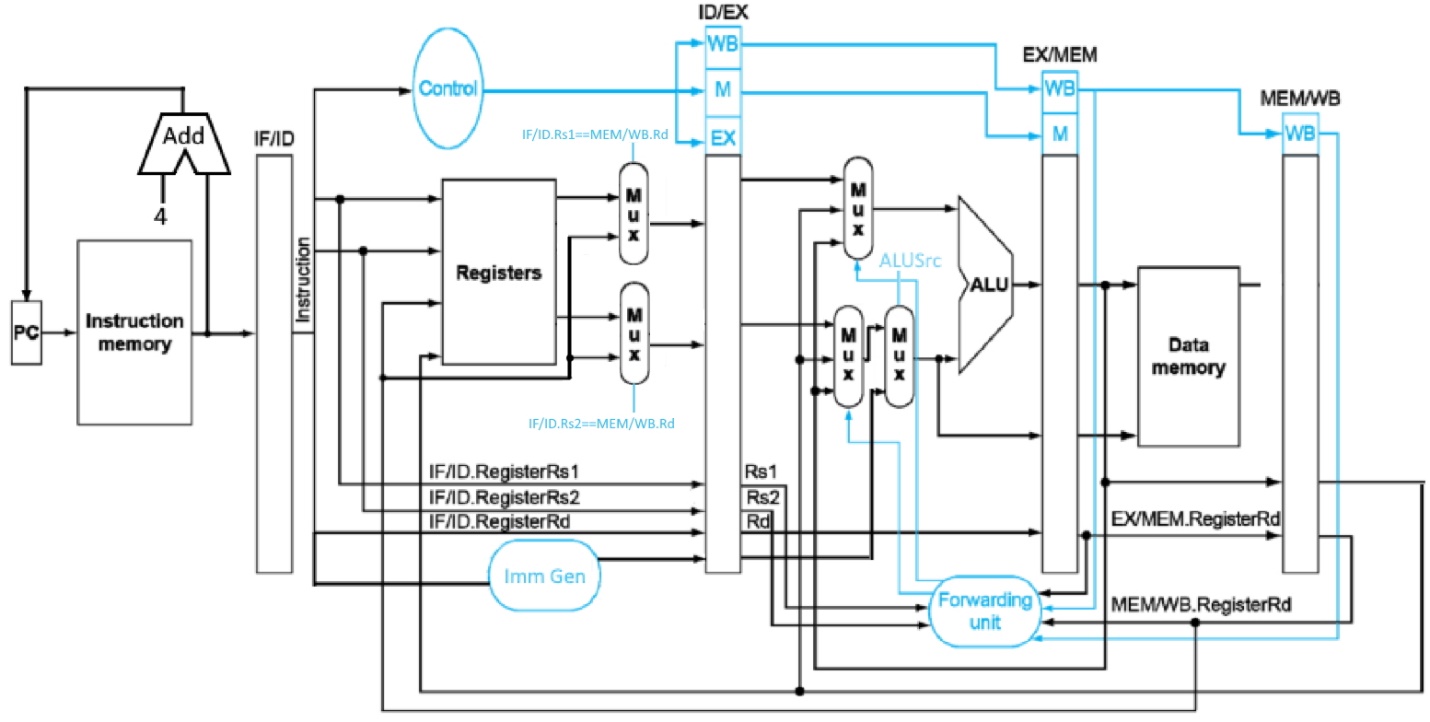
**Computer Organization**

**0716214 江岳勳**

**0716222 黃偉傑**

**Architecture diagram:**

****

**Detailed description of the implementation:**

由於這次Lab承接Lab4，因此我們將只會提及有所更動的實作內容。我們在確認這次的Lab的需求後，決定將許多與load, store, branch等指令相關的物件拔除，使我們的架構簡化。這次Lab相較於Lab4，新增的module有：Pipeline Register (IF\_ID, ID\_EX, EX\_MEM, MEM\_WB), FowardingUnit。以下我們將從三個角度來說明我們的實作，分別是Pipeline Register、FowardingUnit、以及Pipeline\_CPU(Top Module)。

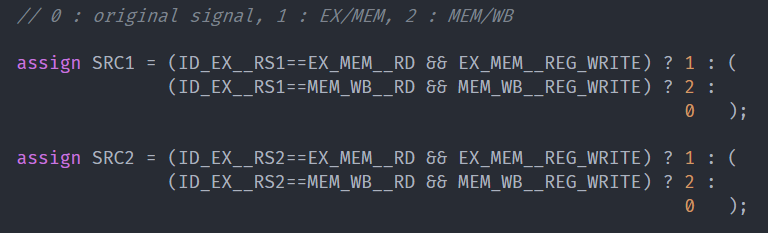
* Pipeline Register

我們是將每個Pipeline Register各自組成一個module，並且控制其在clock正緣觸發時寫入。在省略系統必要訊號與後，剩下的各個Port如下（各有其輸出／入）：

|  |  |
| --- | --- |
| IF\_ID | Instruction |
| ID\_EX | RegWrite, ALU\_Src, ALU\_Ctrl, RS\_Data, RT\_Data,  Imm, RS\_no, RT\_no, RD |
| EX\_MEM | RegWrite, ALU\_Result, RD |
| MEM\_WB | RegWrite, ALU\_Result, RD |

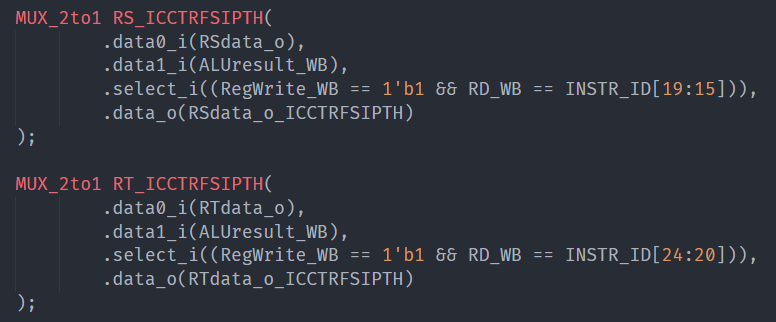
其中與Slides不同之處，除了將用不到的Port拔除以外，由於Instruction被我們從ID\_EX與其後面的Pipeline Register中拔除，因此我們也同時將ALU\_Ctrl module從EX stage往前挪至ID stage，連帶所影響的，是原先ID\_EX中應該要連接的ALUOp改成了ALU\_Ctrl。

* FowardingUnit

這個module的邏輯與Slides完全相符，實作上我們則利用了三元條件運算子，如圖。

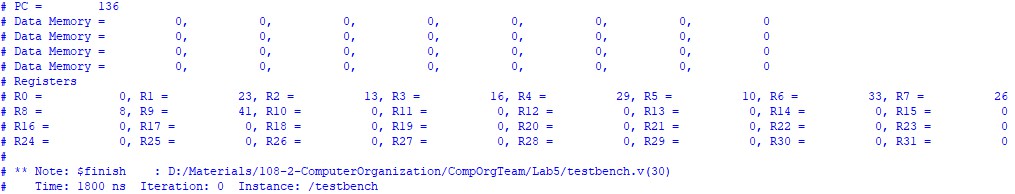
* Pipeline\_CPU (Top Module)

1. 我們將上述的Pipeline Register與Forwarding Unit加入Module之後，便按照我們所學過的5-stage 順序將所有submodule按照順序排列，並且宣告了許多wire來滿足新的module的port。
2. 在Forwarding Unit附近，我們模仿了Slides上在ALU前面接了兩個3-to-1 MUX的作法，而原先用來選擇RT\_Data與Imm的2-to-1 MUX被放在了第二個3-to-1 MUX的後方。
3. 在RegFile與ID/EX中間，我們接上了兩個2-to-1 MUX，用來forward從WB stage寫回來的資料，其判斷條件是WB stage的RegWrite為1且其RD與ID stage的RS或RT相等，如圖。

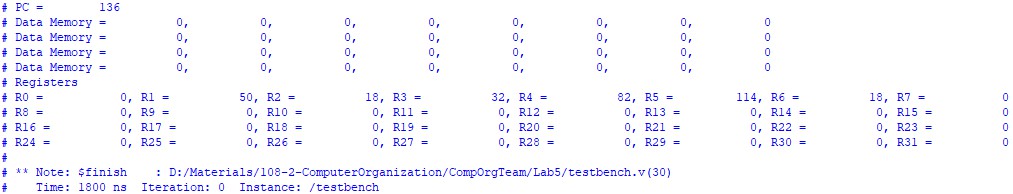


**Implementation results:**

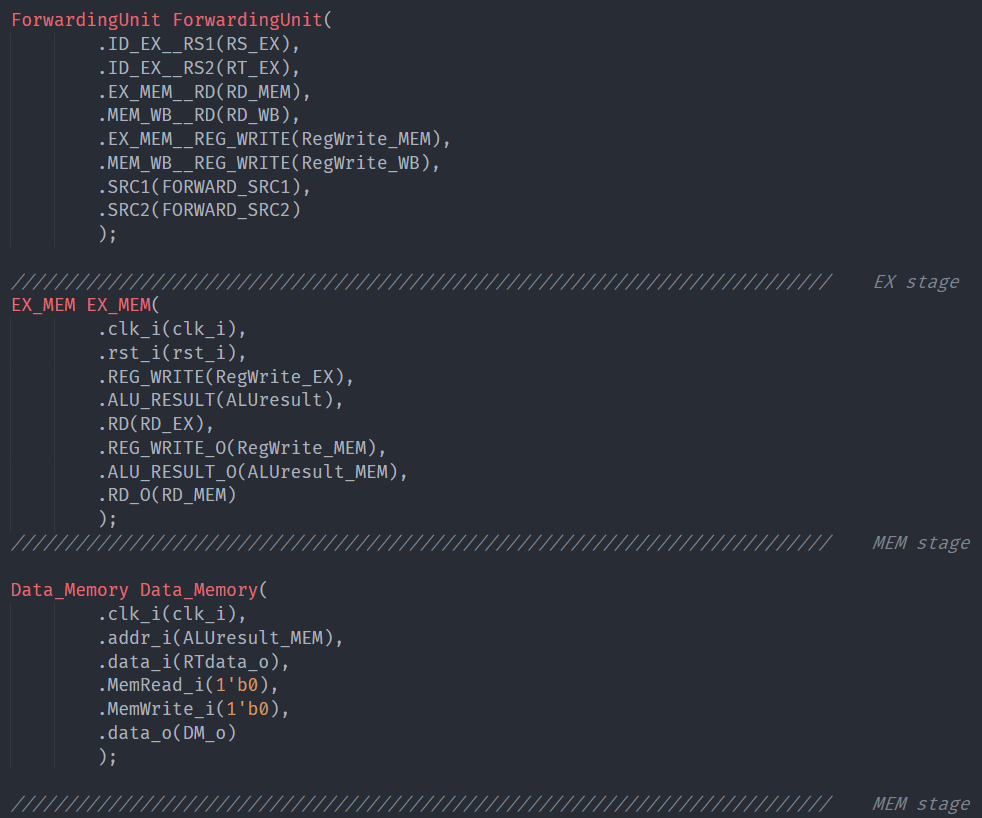
CO\_Data\_1.txt

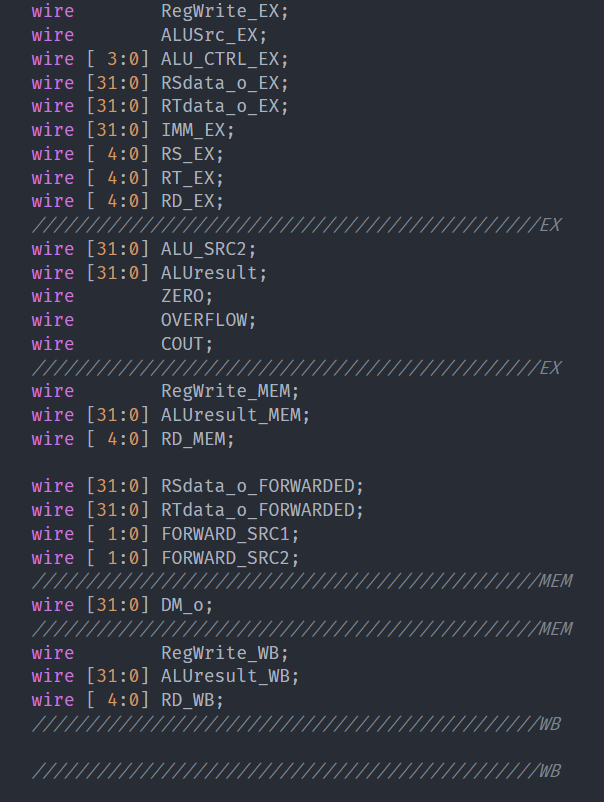


CO\_Data\_2.txt



**Problems encountered and solutions:**

1. 我們在學習pipelining時所學習的Architecture，若是直接apply在這次Lab上，將會發生一個Hazard：在同一個Cycle內寫入並讀取RegFile，會發生讀取到舊值的情形
   * 我們在RegFile的讀取處加上了兩個MUX，若是發生上述的情形時，則會將寫入值直接Forward，詳細的實作在實作解釋的環節有說明
2. 我們在寫這次Lab時，從Lab4的檔案複製過來，但發現像CPU這種結構複雜的module，很容易讓人看得眼花撩亂
   * 我們按照Pipelining中Stage的順序將Module與自定義的Wire排列好，讓我們在編寫與除錯上能夠對變數有更高的掌握，如圖



**Comment:**

**0716214 江岳勳：**

這次的作業實作了上課講了非常久的Pipeline，在上課的時候學Pipeline聽到都產生恐懼了，不過還好這次的作業沒有要實作Branch, Jump和load/store的指令，瞬間變得非常簡單，只要將線正確的接上，然後注意Forward就可以了。

比較值得注意的是因為RegFile是posedge寫入，在WB要寫回時如果剛好有instruction要讀出來，會不受ForwardingUnit的控制讀到寫入前的資料，所以我們加了一個自己寫的module進去做區分，其實就只是加上兩個MUX而已。

**0716222 黃偉傑：**

這次的Lab比我預想的來的輕鬆，或許是因為不需要處理Load use這類令人頭痛的問題，在學習了各式各樣既有Architecture下的問題與其解決方案後，感覺讓我們自己處理如這次RegFile的Hazard簡直是熟門熟路。很感謝助教這一連串的Lab，讓我們由淺入深的累積CPU的架構知識，即使到了這堂課的尾聲我們仍僅能看到CO的皮毛，但實在是收穫甚多！感謝助教、教授，也感謝這幾次Lab的隊友！