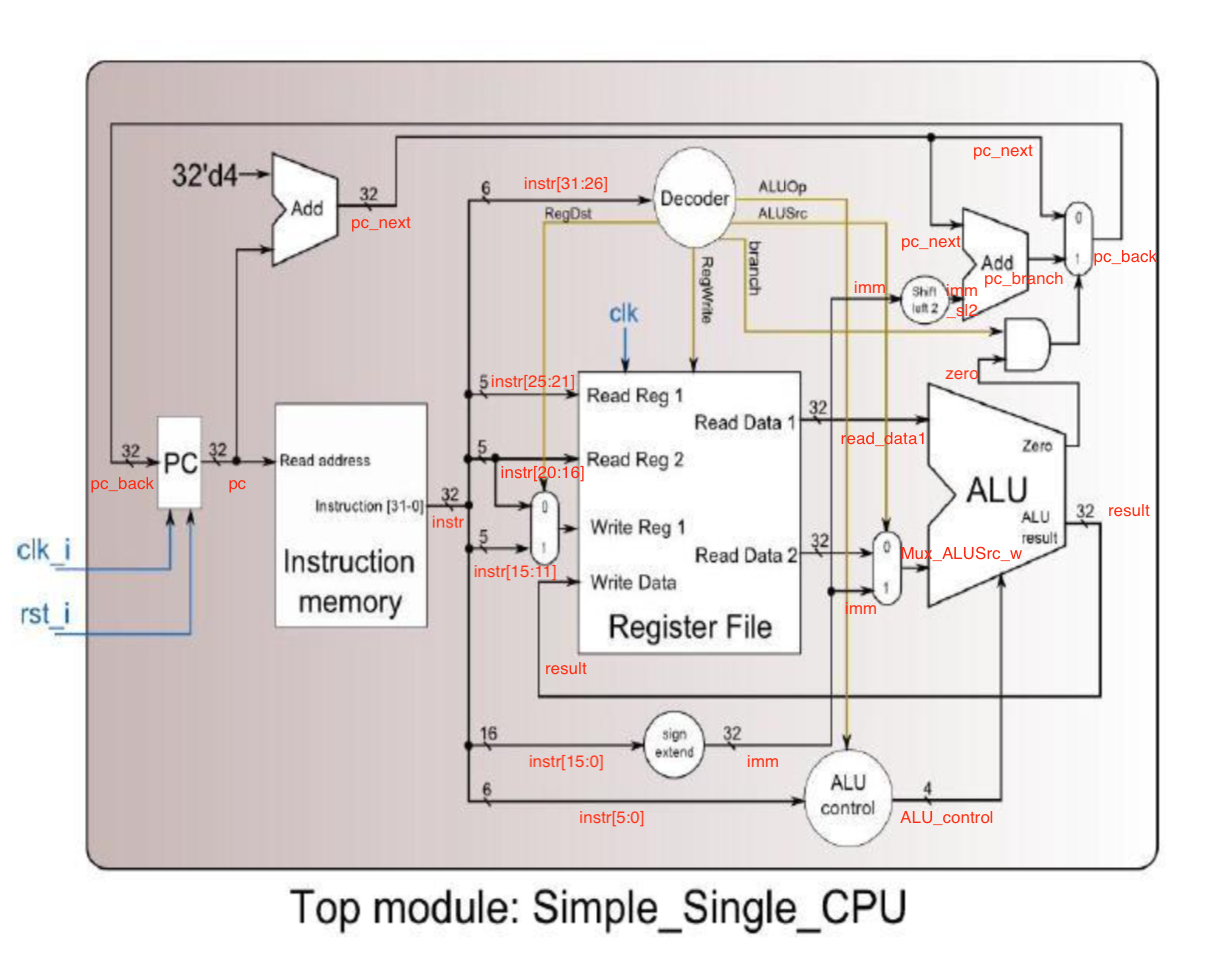
**Computer Organization**

**0516022 吳禹璇**

Architecture diagrams:



Hardware module analysis:

ALU.v我是根據lab2重新寫一份32 bit的ALU。ProgramCounter.v , Instru\_Memory.v , Reg\_File.v是直接使用助教給的，沒有再另外更改。

Adder.v部分，只是做address的加法，只需要把兩個input相加。

Sign\_Extend.v 是把16 bit的imm變成32 bit。

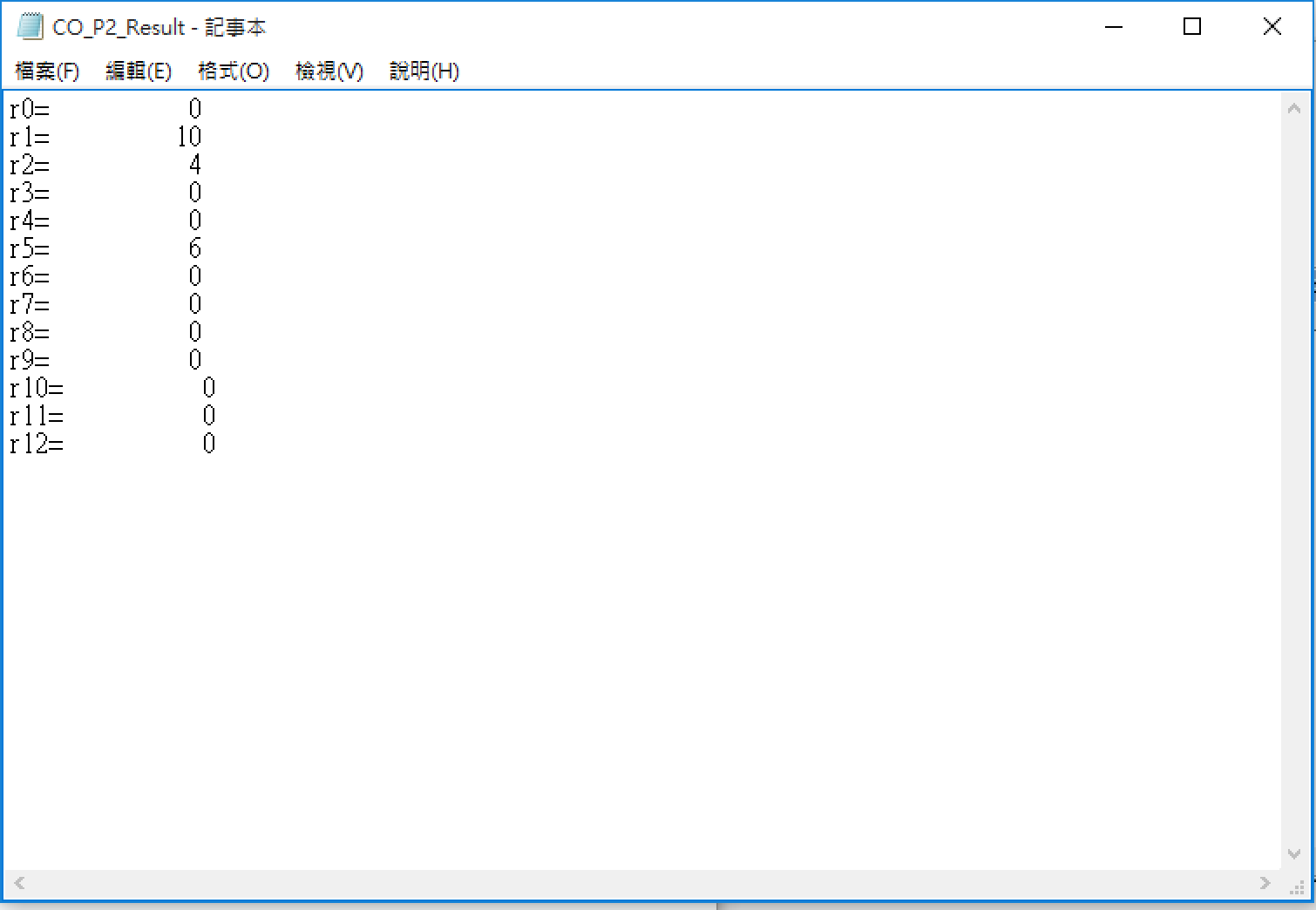
Shift\_Left\_Two\_32.v，因為一開始存到instruction的時候，為了可以存更大的值，有先將address往右移兩位，省略重複的兩個bit，這邊是要把它還原回來而已。

MUX\_2to1.v這部分是以Decoder.v跑出來的control signal作為select的條件，輸出我們要做的operation。這次的設計中有三個地方用到，分別是Mux\_Write\_Reg，根據instruction來判斷最後result要寫在哪個register，Mux\_ALUSrc，決定用instruction裡面的imm還是reg的data來當作alu的input，Mux\_PC\_Source則是決定branch是否成立，若成立，用adder再算一次新的address傳回pc。

ALU\_Ctrl.v根據Decoder.v跑出來的ALUop決定alu的ALU\_control。

Decoder.v 將instruction的op code decode出好幾個control unit，這邊因為spec中沒有給sign extend的signal，我還是把它加上去了。

最後是Simple\_Single\_CPU.v，主要是要把前面提到的module拼湊起來，我設的變數名稱有標在上面architecture diagrams中方便在接線的時候對照。

Finished part:



Problems you met and solutions:

這次作業本來想延用第一次lab做的ALU，但是實作後發現，若用上次做出來的alu，r0會跟著亂跑，原因是因為，這次lab的alu是不被clk及rst控制的，所以我用lab2給的code，直接重新寫了一份32 bit的ALU。

另外，在最後Simple\_Single\_CPU.v要做接線的時候，還是很容易混淆或是一時忘記要怎麼接，後來就先把變數標在圖上，後來也就比較順利的把線都接起來了。

Summary:

相較於上次lab，我覺得我更熟悉verilog的語法了，加上這次lab我有提前開始做，所以都比較順利，唯獨在clk的使用上還有點不熟悉，不過經過這次的經驗，希望下次會更順利。