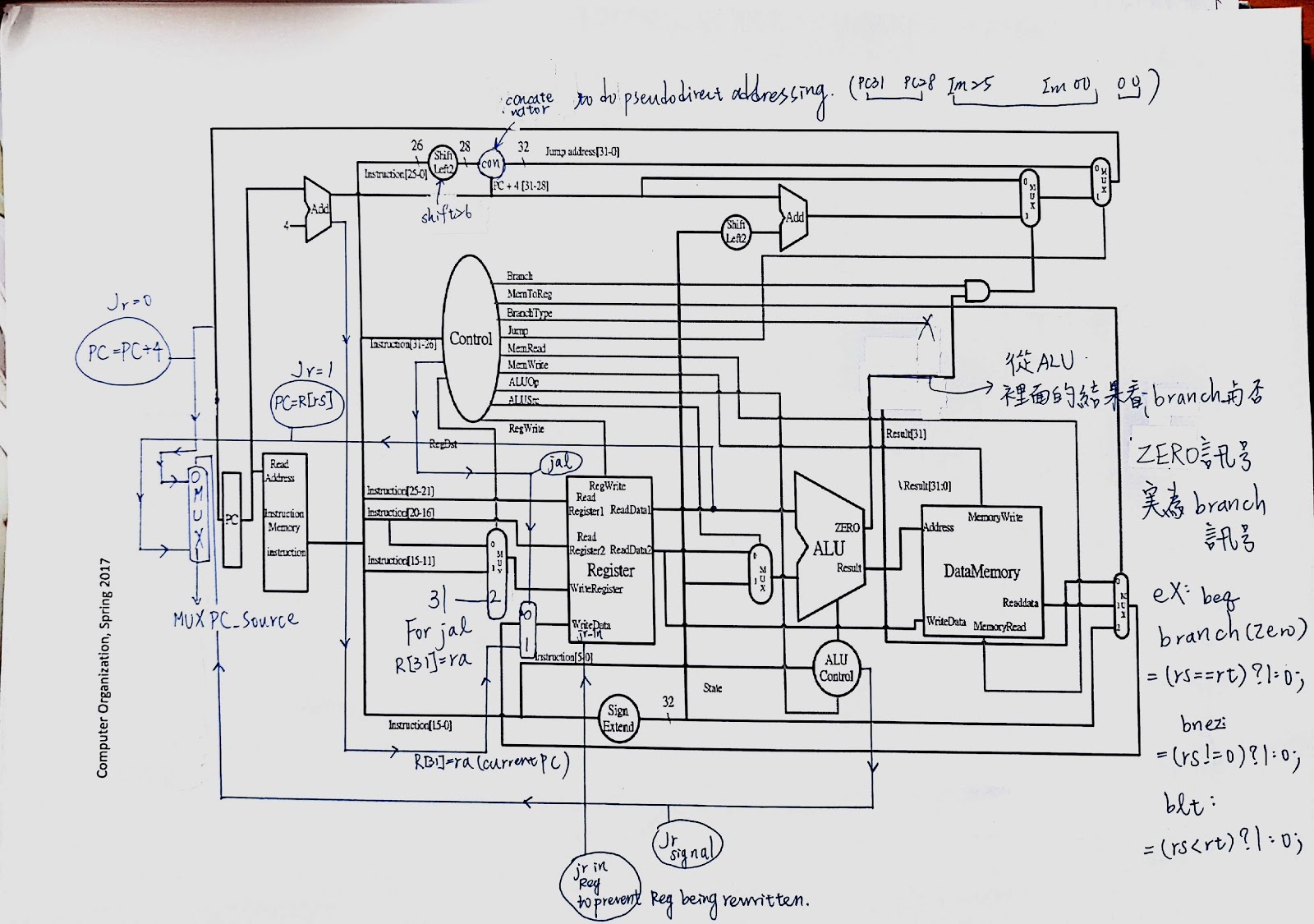
Computer Organization

0416324胡安鳳 0416094黃兆宇

Architecture diagram:



Detailed description of the implementation:

我們從原本的架構去進行修改，加上了

（從左到右）

1.為了處理jr的情形，首先加上了一個MUXPC\_Source 用來決定是否要將jr (jump return)進行PC=R[rs]的行為，由於jr是屬於r type的指令集，因此在解碼器階段上無法看到這個東西是否為一個jr (因為對於r type的指令集前方的op code都會是零，無法判斷）要等到ALU\_Ctrl的時候才能知道利用function code去判斷，在此街一個jr出來給MUXPC\_Source，來決定說要選那一個訊號原作為

程式計數器的來源。

2.在解碼器的部份額外新增了jal的訊號（因為他是j type在進到解碼器的時候就可以做判定），將訊號接給Reg File前面的4 to 1 MUX\_2to1 Mux\_Write\_Data\_Select以選擇說是否要將當前的程式計數器存進$ra中，還額外將regdst改為2bit的形式作為選擇RegFile register address的來源，連接到同樣位於RegFlie左方的  
MUX\_4to1 Mux\_Write\_Reg\_Select，看是否要進行Reg[31]的選擇來儲存當前的程式計數器或只是單純的依據i r type的不同來決定說是要寫入rs 還是rd。

3.ALU\_Ctrl的jr訊號除了接給上方第一點所說的MUXPC\_Source

作為選擇當前的程式計數器以外還要接給RegFile，原因乃是RegFile在解碼器解析出r type的指令集的時候會進行暫存器的寫入，然而jr也是r type指令集卻不進行暫存器的寫入，如此一來可能會造成有雜訊進來暫存器裡面的情形，會使得結果出錯。

4.最後我們修改原本架構圖右上方複雜的BranchType所連接到的電路，改將ALU的Zero作為branch訊號，方法很簡單

always@(\*)

begin

if(rst\_n)

begin

if(ctrl\_i==4'd9)//BEQ

zero\_o = (result\_o == 0) ? 1 : 0 ;

else if(ctrl\_i==4'd10)//BNE

zero\_o = (result\_o == 0) ? 0 : 1 ;

       else if(ctrl\_i==15) //BLE

           zero\_o = (src1\_i<=src2\_i) ? 1 : 0 ;

       else if(ctrl\_i==16) //BLT

           zero\_o = (src1\_i<src2\_i) ? 1 : 0 ;

       else if(ctrl\_i==15) //BNEZ

           zero\_o = (src1\_i!=0) ? 1 : 0 ;

else

zero\_o = 0;

end

else

zero\_o = 0;

end

就省去了麻煩的電路了。

以上便是架構的實做

Problems encoered and

1.接線時，會不小心遇到size mismatch 的問題。

2.執行jr, Rs 從錯誤的地方接出至pc，應該在reg file後得到address再將其接出，而非輸入reg file 之前。

3.Jr 時應其是r type而將reg write設為1 ，造成錯誤。

lesson learnt (if any):

1.了解各種j type的接法，還有接線時須注意的細節。

2.了解jump, branch 和其他instruction之間切換的機制。

3.了解mips遞迴程式的寫法。(使用jr jal)