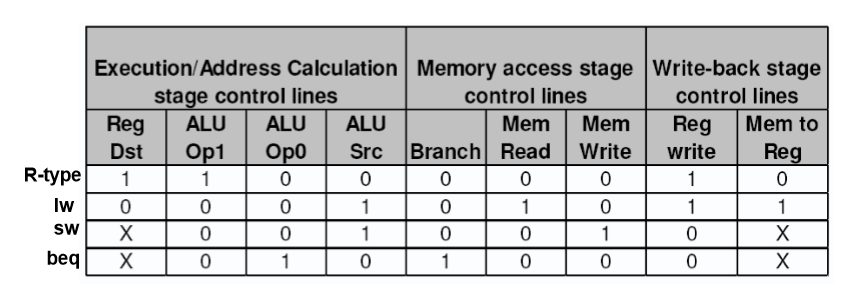
101502014 資工2A 鄭俊廷

一開始先利用fstream函式庫中的ifstream讀入文件中的machine code，再分析machine code的opcode和function判斷出是哪個指令，再將那個指令的rs , rt , rd或immediately記錄下來。



依照講義上的圖決定control singnal，先以十進位存入，輸出時再利bitset函式庫中的”bitset<位數>變數名稱(control signal)”，輸出指定n-bit的二進位數字。

首先，為了要解決Data Hazard，要先算出ForwardingA跟ForwardingB，判斷是否要先將後面階段(EX/MEM)或(MEM/WB)尚未寫入暫存器或記憶體的值提早拿到第二個階段(ID/EX)使用，先將第一個指令丟到pipeline的第一個階段(IF/ID)，輸出目前的PC值(一開始為1)和指令的machine code。

第二個階段(ID/EX)則輸出ReadData1($rs)和ReadData2($rt) , sign\_ext(immediately或是0(R-type))和Rs , Rt , Rd和Control signals(9-bit)。

第三個階段(EX/MEM)輸出ALUout和WriteData(指令是sw則輸出$rt，否則則輸出0)和Rt(lw , sw , beq)或Rd(R-type) 和Control signals(5-bit)。

第四個階段(MEM/WB)輸出ReadData(指令是lw則輸出mem[ALUout]，否則輸出0)和ALUout(指令是sw則輸出0，否則輸出ALUout)和Control signals(2-bit)。

每個階段輸出完後，從最後一階段開始丟出指令，第四個階段(MEM/WB)把ALUout寫入暫存器，另外$0永遠為0。

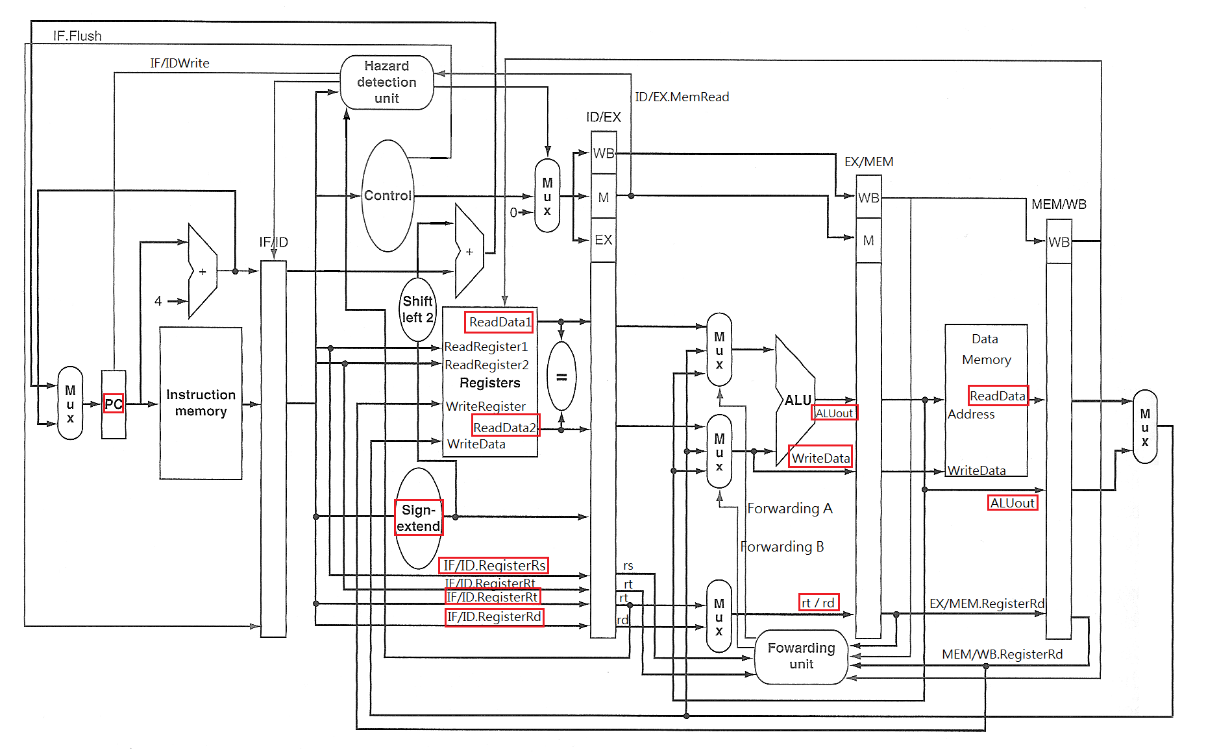
第三個階段(EX/MEM)是寫入記憶體(sw)或讀取記憶體(lw)，如果是R-type，則把ALUout沿用到下個階段去寫入暫存器。

第二個階段(ID/EX)用來進行$rs跟$rt的運算(R-type)、immediately的運算存入ALUout(lw、sw)，還有beq遇到的Branch Hazard，解決方式是將PC值加上immediately-1，並把第一個階段(ID/EX)的control signal歸0，讓讀入的指令序列提早跳到beq指令的位址。

最後在進入下一個cycle之前，要把前三個階段的指令資料和control singals丟到下一個階段，並將還沒丟進pipeline的指令丟到第一個階段，PC值加一，還有Load Hazard的判定，解決方法是Stall，將第一個階段(ID/EX)的指令資料和control signals歸0，並讓cycle繼續進行。

所有cycle結束的判定是直到所有指令都在pipeline中執行完成，故就判定每個階段的指令是否為空。

下圖為寫這pipeline參考的圖片：



部分邏輯與想法參考這個部落格：

http://mypaper.pchome.com.tw/zerojudge/post/1324734182