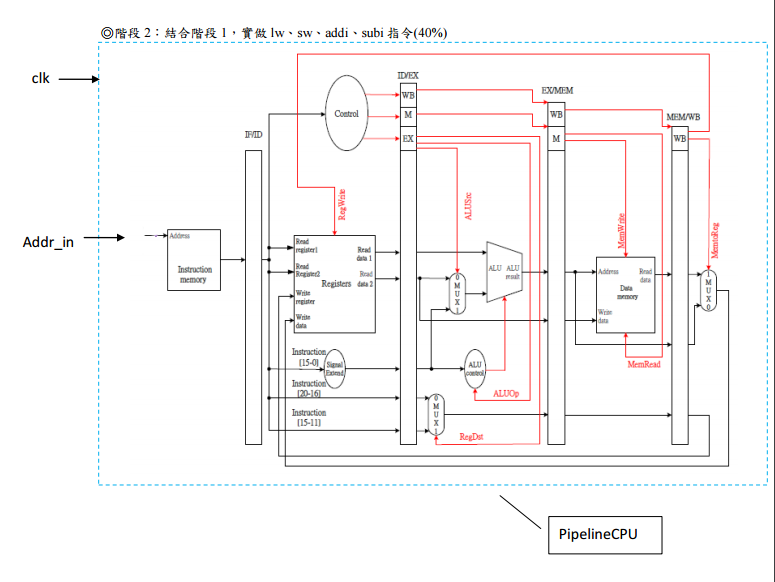
計算機組織

* 1. 架構



* 1. 題目說明

階段一:

完成ADD SUB AND 以及OR 四種功能

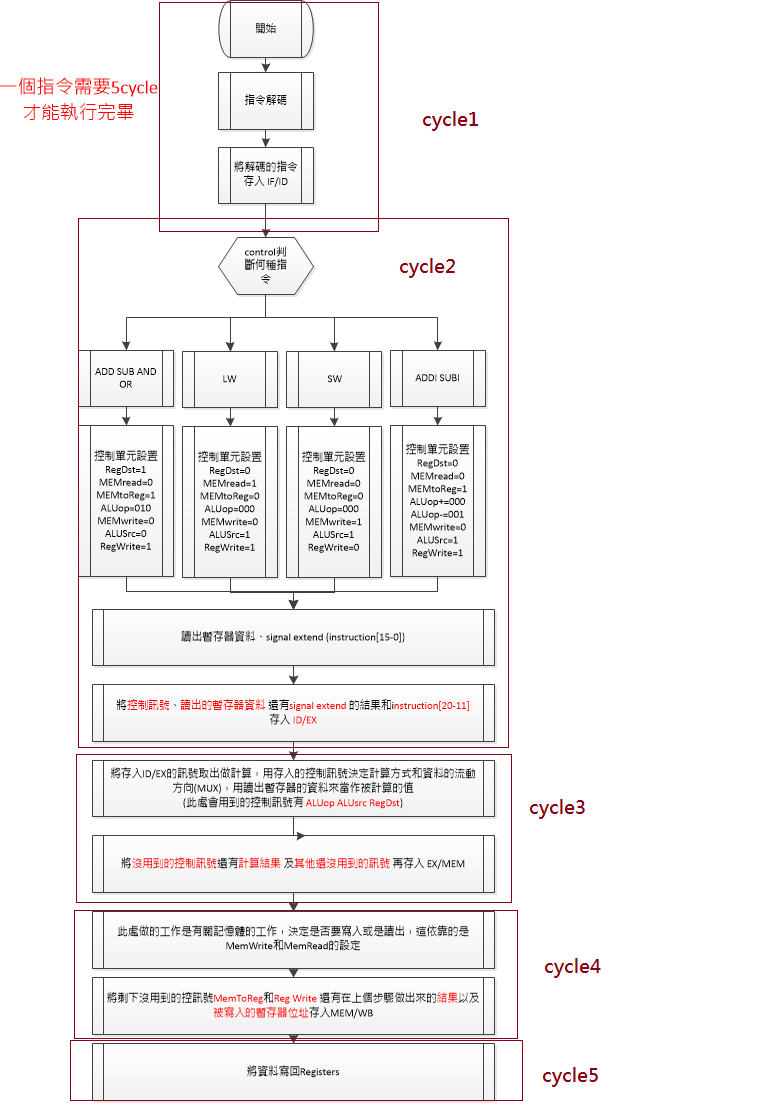
階段二:

實作 LW SW addi subi指令，

階段三:

實作 J-type 或 forwarding 以及Datahazqrd (沒完成

* 1. 流程圖



* 1. 程式說明

此題目的基本元件control、register、ALU、ALUcontrol、DATA MEMORY基本上和之前的project 沒有太大的變化，只有在data memory的地方我把它改成了負緣寫入，沒有正緣的部分。而變化最大的就是instruction memory的地方，因為指令會產生data hazard，所以我作了reordering的技巧改善這個問題

1. Data Hazard：下一個指令所需使用的資料，還無法獲得時，也就是資料相依的問題。在這處理器中容易產生的Data Hazard的原因是因為register讀取和寫入的時間是不相同的，讀取是在最初的時候，而寫入是在最後的地方，因此如果第二個指令要用低一個指令的結果就會用不到新的資料，只能取到舊的資料造成結果不同

處理方法

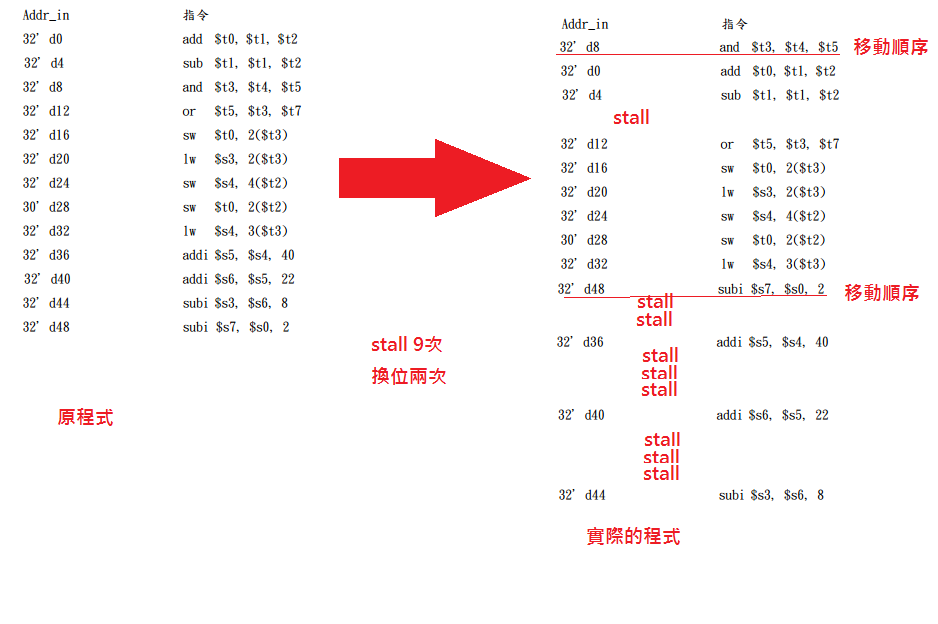
1. 方法一：加入Bubble （類似Delay的概念）
2. 方法二：用Data 之 ByPass or Forwarding前餽的方式。也就繞過Register先將資料傳給要使用此結果的單元。
3. 在不改變程式的結過下，也可使用指令移動順序的方式

我使用的方法是3和1

在這些指令中會發生data hazard的地方有8->12->16->20的地方，12 20和16都需要用到$t3的結果，可是她們拿到的資料卻是舊的$t3的資料，所以做出來的結果會有問題

而在32->36->40->44的地方也會發生data hazard， 36會用到32的結果，40會用到36的結果，44會用到40的結果，所以都要解決data hazard的問題

解決方式



在指令0~16會發現指令相依的有 8和12、16、20，8的結果12 16 20都會用到

，以我們的processor，一格指令從輸入到要改變REG需要等待3個cycle的時間才能寫入成功，因次8之後的後三個都不能執行成功，所以8需要提早完成，因此我們把8提到最先作的地方然後0和4往後移一個，接下來再STALL一次，就能成功空三格，改變REG

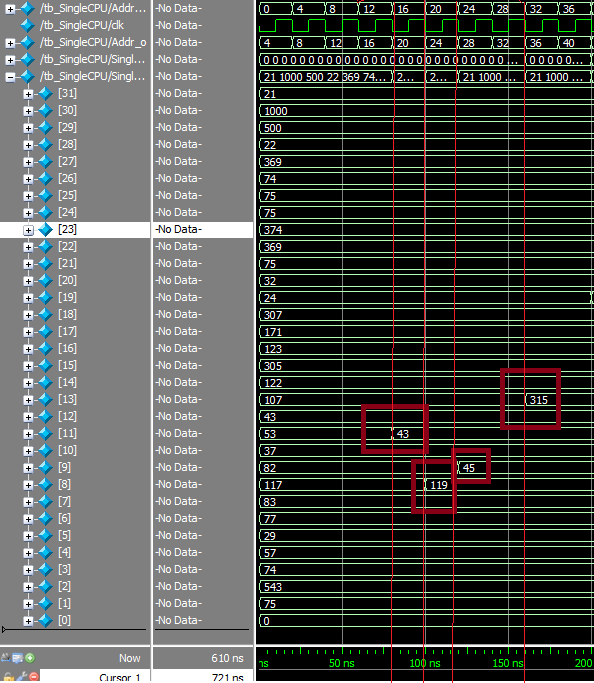
在20~28會發現每個指令都和前一個有關，因此每個指令幾乎到要stall三次，這無法避免，所以從32後要STALL三次 ，36後要STALL三次 ，

40後要STALL三次，而48的指令和前面都無關所以可以在前面的stall完成，此時就可減少1格stall ，共stall八次

這兩格部分總共stall 9次

* 1. 結果

暫存器1



根據移位結果 8會先作，所以會先得到43的結果

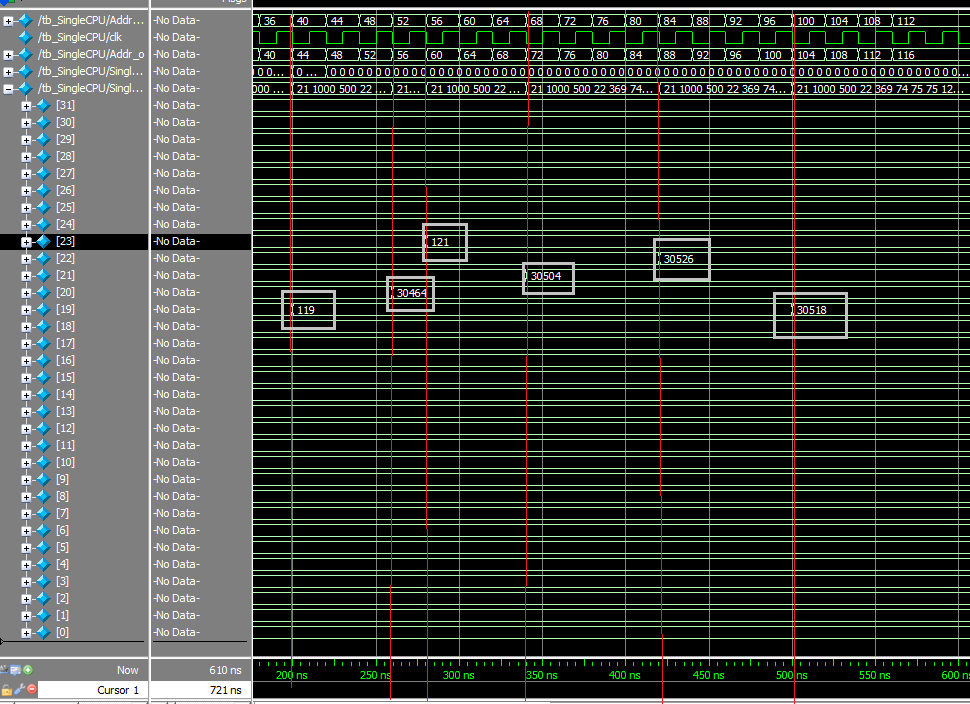
接著是 0相加得到119

4，相減得到45

然後stall一次

在執行 12得到315

暫存器2

****

接著的指令是 16 存入，所以暫存器沒變化(過3cycle才寫入)

20寫入，暫存器將119存入暫存器中

24存入，暫存器沒變化

28存入，暫存器沒變化

32寫入，暫存器寫入30464(過3cycle才寫入)

48 指令先做，不影響結果，減法得到121

stall一次

stall一次

36加法，+40得到30504

stall一次

stall一次

stall一次

40加法，+22得到30526

stall一次

stall一次

stall一次

44減法，-8得到30518

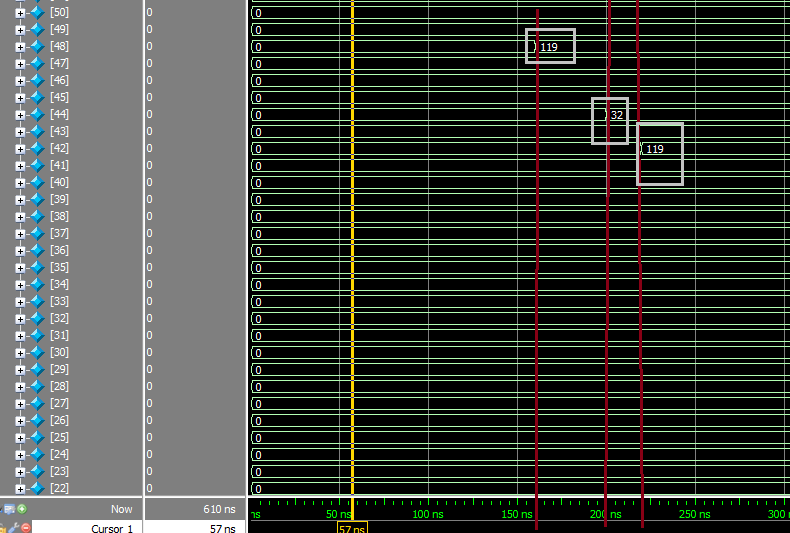
stall一次

stall一次

stall一次

存回REG

**記憶體**

****

16 存入，記憶體存入資料(過3cycle才寫入)

20寫入，暫存器將119存入暫存器中

24存入，記憶體存入資料(過3cycle才寫入)

28存入，記憶體存入資料(過3cycle才寫入)

心得

時間太趕來不及作階段三，但希望之後比較清閒的時候再把階段三做好，這學期學了很多大概了解了微處理機的基本原理，能用程式實作一格東西的感覺很好