**專題二報告**

課程主題： 計算機結構

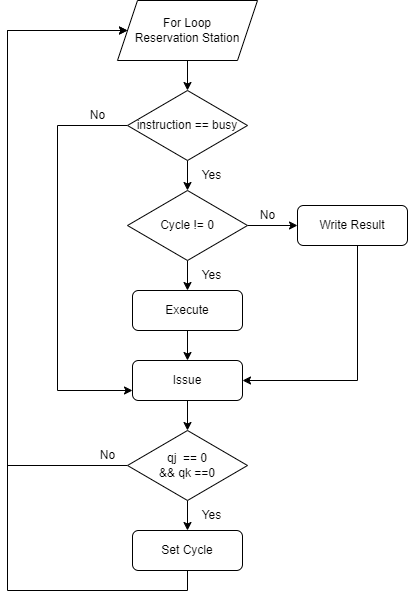
學號：A1095551

姓名：廖怡誠

本次專題的內容為設計Tomasulo algorithm的模擬器，根據題目的需求設置Reservation Station、Register Result status、Cycle time等參數，並且能夠避免RAW、WAW、WAR三種Hazard。

1. **架構設計**

Tomasulo algorithm所使用的架構較多，如Reservation Station、Register Result Status、Register、Memory等，此外，還需要管理進入instruction queue的指令，所以我分別建立Reservation Station、Register Result Status、instruction unit與Tomasulo四種Class，目的是後續在建立陣列時，方便管理與呼叫使用。



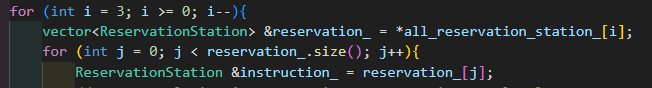
**圖一、Tomasulo Algorithm的流程圖**

圖一為Tomasulo Algorithm的流程圖，1)首先，尋訪Reservation Station中的Adder、Multipliers、Load buffer、Store buffer是否有Busy狀態的指令，2)接著，判斷該指令的執行cycle是否已經完成，如果cycle為0，則代表執行完成，執行Write Result階段；反之則會執行Execute階段，並更新cycle的數值，不同類型的指令會執行對應的動作，3)結束Reservation Station的尋訪之後，再進入Issue階段，從Instruction queue中取出instruction，並設定後續判斷需要使用的參數，4)最後，會檢查放入Reservation Station的指令是否有Qj與Qk均為0，代表能夠進入Execute階段，因此設定對應的Cycle值。

由於優先執行Issue之後，會出現同一個Cycle內，剛放入Reservation Station的指令會進入Execute，因此需要將Execute與Write Result的順序提前，且一指令在相同的cycle中不會同時進入Execute與Write Result，此外，Write Result的Reservation Station會等到Issue結束後才執行初始化，避免在相同的Cycle中Write Result的硬體空間，馬上被使用。

1. **遭遇問題**
   1. **Store指令Write Result時機**

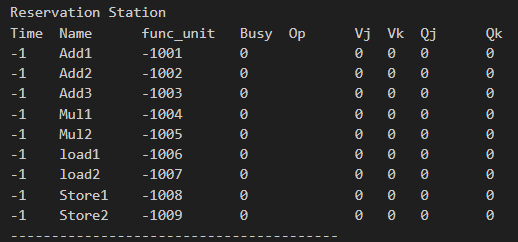
Store指令進入Write Result階段的條件與其他指令不太相同，當RS[r].Qk為0且執行完成才會進入Write Result，所以如果根據上述的執行順序(Adder、Multipliers、Load buffer、Store buffer)，Store 指令可能會因為前面的指令Write Result使Store的Qk變成0，導致在同個Cycle會直接進入Write Result階段，但由於CDB傳送Data需要花費1個cycle，因此實際上Store 指令要到下個Cycle才可以進入Write Result階段，為了避免發生這個問題，我將執行順序反轉變成Store buffer、Load buffer、Multipliers、Adder，因為優先執行Store buffer就不會提早判斷。



**圖二、執行順序反轉**

* 1. **Function Unit管理**

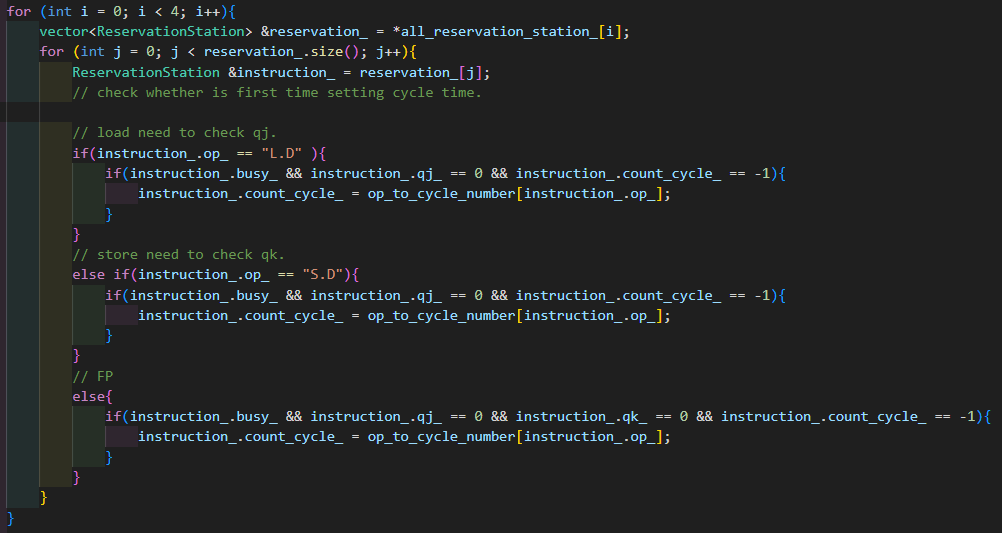
原先在書面推導cycle時，可以用看得確認每個指令對應到Reservation Station與Register Result Status的位置，但實際以程式實作時，除了需要管理instruction queue的執行順序與其對應到的位置，還要記錄該指令以便彼此之間溝通，例如，記錄cycle需要找到對應的指令和Reservation Station要在Register Result Status中找對應的指令存值，因此，我將Reservation Station的每個位置都加入一個編號，方便在instruction queue與Register result status中找出對應的指令，此部分是書面推導不會遇到的問題，也不太確定實際上的運作會是如何運作。



**圖三、設置Reservation Station的func\_unit數值**

* 1. **Reservation Station 初始化問題**

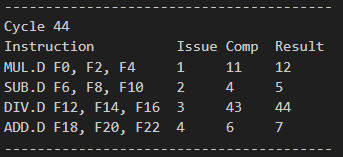
在Reservation Station的初始化中，以硬體架構來說，沒有值的時候，該位置的值應該要為NULL，然而在實作中，由於要避免判斷時，讀取到NULL出現問題，且在C中無法給一個變數NULL的空值，因此只能都將其設值為0，然而，這樣的處理方式會與判斷進入Execute階段雷同，所以我在判斷是否要進入Execute階段中，會加入該指令是否於busy與其cycle是否被設置，將沒有使用到的部分加入cycle。



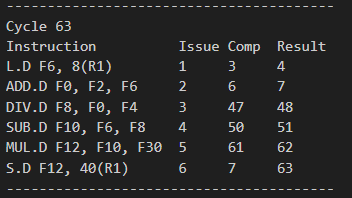
**圖四、加入額外判斷參數**

1. **執行結果**

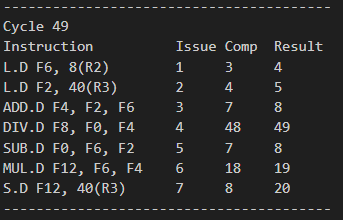
Test1 result



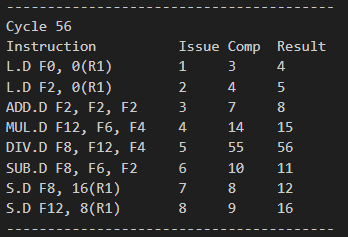
Test2 result



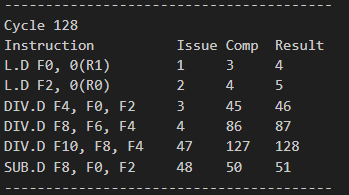
Test3 result



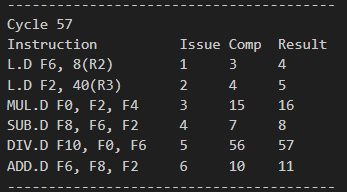
Test4 result



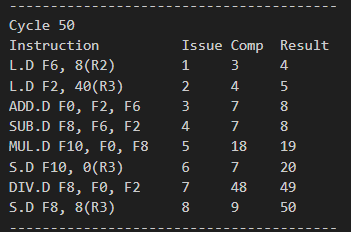
Test5 result



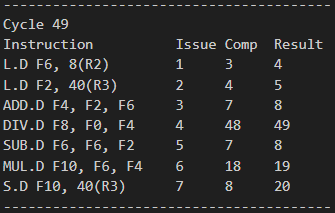
Sample1 result

****

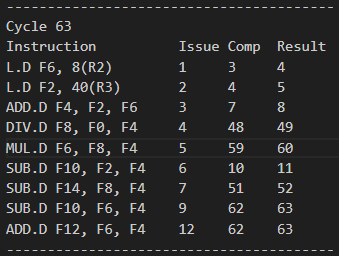
Sample2 result



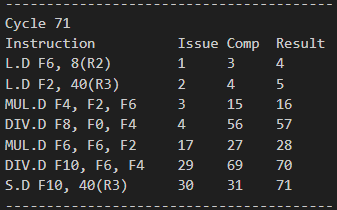
Sample3 result



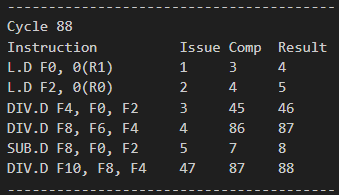
Sample4 result



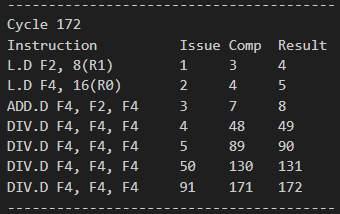
Sample5 result



Sample6 result



Sample7 result

****

1. **心得**

在這次的實作中，實作了Tomasulo Algorithm，了解到使用軟體模擬硬體的實際架構與程序仍然有一定程度的困難，需要考慮到執行操作通常是1個cycle為單位、或是硬體的線路傳資料使用軟體只能使用指標和標籤來代替，這些問題使我在實作的當下十分糾結，會好奇實際的情況會如何運作，如果用軟體模擬用甚麼樣的寫法能夠更接近實際的硬體運作，但兩者還是有差別，沒辦法實作到完美。除此之外，參照投影片上的Issue、Execute、Write Result三個流程的說明也十分有挑戰性，感覺像是一個人要實作Pipeline的複雜度，一步一步跟著說明實作，並發現其中的Store與Load不會有到Register Result Status確認Qi的步驟，因為通常Store與Load是到Memory拿資料，但是Register Result Status紀錄的Double型態的Register，如果直接取值，會獲取並存在的位置。