|  |
| --- |
| **Tomasulo演算法** |
| **Input:**  *FileName* //檔名  **Output:**  *Total Clock* //執行全部指令總時間週期  **Method:**  **Step 1:** 初始化參數  **Step 1.1:**  Reservation Station   1. *loadBuffer*、*storeBuffer*、*adder*、*multiplier*，分別有2、2、3、2個資源。   **Step 1.2:** Register Status   1. 浮點數暫存器有16個，編號為F0、F2、F4、…、F30，初始值為1。 2. 整數暫存器有32個，編號為R0、R1、…、R31，除R1的初始值為16外，其餘整數暫存器初始值為0。   **Step 1.2:** 指令執行週期   1. L.D=2、S.D=1、MUL.D=10、ADD.D=2、DIV.D=40。   **Step 2:** 建立子類別  **Step 2.1:** Instruction class   1. *issue* //表示第幾個clock有資源可用 2. *execution* //表示開始執行的clock 3. *executed*(execution complete) //表示執行完畢clock 4. *written*(write result) //表示被寫入clock   **Step 2.2:** Reservation Station   1. *id* //表示第幾個指令(行數) 2. *busy* // 0:閒置中 1:資源佔用中 3. *opcode* // operand指令種類 4. *Vj, Vk* //可以馬上使用的實際數值 5. *Qj, Qk* //目標暫存器的值還在處理中等待廣播 6. *remain* //儲存還剩餘幾個cycle將結束執行 7. *flush()* //清空保留站function   **Step 2:** 執行tomasulo演算法  **Step 2.1:** WriteResult執行已完成Execution步驟的指令  **Step 2.1.1:** 檢查全部的保留站，是否已經可以寫入結果  **Step 2.1.2:** 若已經執行結束的指令，將會廣播更新保留站的Qj或Qk並寫入結果  **Step 2.1.3:** 清空已經執行完畢的保留站並釋放空間  **Step 2.2:** Execute執行已完成Issue步驟的指令，並在維護每個正在本步驟的指令  **Step 2.2.1:** 檢查所有的保留站，能立即執行的就開始執行  **Step 2.2.2:** 檢查指令是否可以結束執行  **Step 2.3:** issue下一個未被issue的指令  **Step 2.3.1:** 檢查目前指令是否可以被 Issue  **Step 2.3.2:** 檢查該指令是否有 true dependence 狀況發生  **Step 2.3.2:** 判斷目前是否有保留站可以被使用  **Step 3:** 印出結果並寫出optput.txt  **Step 4:** 結束 |