Report

資工三 蕭千惠
資工三 陳　昇
資工三 陳佳佑

I. Pipeline Structure

將原先 single cycle 的 CPU module 加入了 IFID、IDEX、EXMEM、MEMWB 這 4 個 module，使得程序可分為 5 個階段進行。同時加入了 ForwardUnit 來避免不必要的 stall，以提升整體的運算效率，並且額外設計了 HazardDetection，以實現 lw 指令所造成 data hazard 時，無法避免的 nop。另外，在面對 beq 以及 j 指令所產生的 control hazard，也提供了 flush 的機制來讓我們能夠清除原來錯誤的指令。

II. Modules Implement

1. ALU

   依照 ALU_Control 所給的輸入來決定該進行何種運算，不同的參數分別代表著 OR、AND、加法、減法、乘法等不同的運算模式。

2. ALU_Control

   依照 ALUOp 以及不同的 funtion 來分別決定對應的 ALUCtrl，並將此結果傳至 ALU 作為運算根據。

3. Adder

   將兩輸入相加後輸出。

4. Control

   根據不同指令的 Op code 來分別決定 ALUOp、RegDst、ALUSrc、MemtoReg、RegWrite、MemWrite、MemRead、Branch、Jump 等參數。Branch 送入 Flush 以及 MUX_Add 做運算，Jump 送入 Flush 以及 Mux_Jump 做運算，其餘的參數則送入 MUX_Control 做處理。

5. DataMemory

   建構一個 32*8 bytes 的記憶體，若 MemWrite 為 1，則將資料寫進記憶體；若 MemRead 為 1，則將資料讀出 32 bits 作為輸出。

6. EXMEM

   將 Mem stage 要用的參數傳至對應的結構中。

7. Flush

   分別考慮兩種情形，會啟動 flush 機制。一為 j 指令，若 Control 傳入的 Jump 為 1 時會導致 flush。二為 beq 指令，若 RS 和 RD 檔案相同，並且 Control 傳入的 Branch 為 1 時也會導致 flush。

8. ForwardUnit

   分別考慮兩種情況，會啟動 forward 機制。一為 EX hazard，若 RegWrite 為 1，且 EXMEM 的 RD 和 IDEX 的 RS 或 RT 相同時，則代表相鄰兩指令之間需要 forward。若狀況一並未發生，則考慮狀況二，MEM

hazard。若 RegWrite 為 1，且 MEMWB 的 RD 和 IDEX 的 RS 或 RT 相同時，則代表此兩指令之間需要 forward。並將輸出傳至 MUXforward_1 和 MUXforward_2 做運算。

9. HazardDetection

若 MemRead 為 1，則代表我們要進行 lw 指令。此時倘若 IDEX 的 RT 和 IFID 的 RT 或 RS 相同，則代表著我們必須要 stall 一回合，並且通知 IDEX、PC、MUX_Control。

10. IDEX

將 MUX_Control、Registers、Sign_Extend 傳入的資料傳出至 EXMEM、MUXforward_1、MUXforward_2、ForwardUnit、MUX32、MUX5。

11. IFID

若 flush 或 hazard 參數為 1，則將輸出都設為 0，反之則原封不動的輸出 pc 及 inst。

12. Instruction_Memory

由指令的地址取出指令。

13. MEMWB

將 WB stage 要用的參數傳至對應的結構中。

14. MUX32

若 select 參數為 1，則設定輸出為輸入 2；若為 0，則設為輸入 1。

15. MUX5

若 select 參數為 1，則設定輸出為輸入 2；若為 0，則設為輸入 1。

16. MUXForward

根據 ForwardUnit 所傳入的參數，決定是否要 forward 在 EXMEM 或 MEMWB 的資料作為輸出。

17. MUX_Add

若 Zero 和從 Control 傳入的 Branch 同時為 1 時，代表指令的 Op code 顯示為 beq，同時 RS 和 RD 相同，滿足 beq 的條件，此時則將輸出設為跳轉的目標地址；若否，則將輸出設為 PC+4。

18. MUX_Control

若發生 hazard，則將所有 signal 都設為 0；若沒有，則將所有 signal 合成 EX、M、W 傳出。

19. MUX_Jump

若從 Control 傳入的 Jump 為 1 時，代表發生 j 指令，則將輸出設為目標地址；若否，則將輸出設為 MUX_Add 的運算結果。

20. MUX_Write

若 MemtoReg 為 1，則代表指令為 lw，需要將地址傳回 Register 來寫入資料。

21. PC

hazard_i 會判定是否發生 hazard，若發生時則將下個階段的 PC 改為 0 來達成 stall；若未發生則照常使用 MUX_Jump 所提供的下一個 PC 地址。

22. Register

若 RegWrite 為 1，則代表 Register 要將 RD 的資料寫入 register 當中。

23. ShiftLeft26

將輸出設為輸入擴大位數至 28 位元，低位元補 0。

24. ShiftLeft32

將輸出設為輸入向左做 2 個位元的 logical shift。

25. Sign_Extend

以 input 的最高位來決定擴大位數至 32 位元的細節，若最高位為 1 則在前面補齊 16 個 1；反之，則補齊 16 個 0。

III. Problems & Solutions

1. Equal wrong location

原先在 ALU 內就比較兩個輸入是否相等，並輸出對應的 Zero 參數，此舉會導致數值產生的時機並不正確而發生錯誤。將運算的部分改在 CPU 內操作，讓 Zero 在需要使用時才進行運算，就能改正這個問題。

2. MEMWB wrong clock edge

在 MEMWB 中也要放入 clk_i，才能讓數值有順著 pipeline 的脈絡更新，確保整體的正確性。

3. Testbench clock edge

原先 single cycle 的 testbench 直接放入 pipeline 時會發生錯誤，需要在 clk_i 的 negedge 時更新，才能符合我們的 pipeline 設計。

4. ShiftLeft32、MUX_Add、MUX_JUMP bug

許多細部的接線看似十分簡單，但是不夠熟稔的編程總會在程式的字裡行間埋下大錯而難以察覺，用了 GTKwave 後可以快速釐清問題所在，是每個寫程式的人都該要學會使用的好工具，本組全體同學在此感謝資工系廖教授世偉無私的分享此軟件，能夠在死線前安然做完報告都實質歸功於廖教授，特以此篇幅向我們的永遠的導師、永遠的舵手致敬。

IV. Members & Team Work

Single Cycle: 陳昇、蕭千惠

Multiple Cycle: 蕭千惠、陳佳佑

Outline: 陳昇

Debug: 蕭千惠、陳佳佑

Report: 陳昇