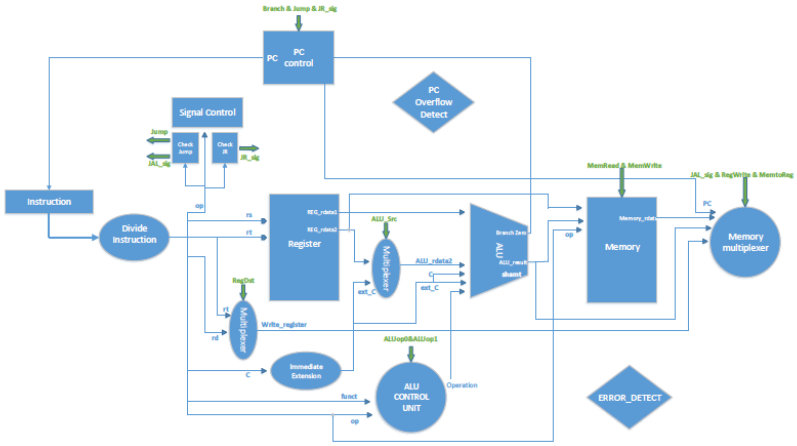
Project1 Report

102070014 廖曼庭

1) Project Description

1-1) Program Flow chart(大圖在附檔~~)



1. singal control 的訊號輸出與課本相同未劃出
2. 簡化訊號傳輸部份，只留下輸出輸入
3. ERROR DETECT 只簡略劃出，詳細功能在(1-2)detail

1-2) Detailed Description

Single_cycle detail(與課本相同處不敘述)

1. 除了課本原先的 signal 外另外加了 JAL 和 JR 的 signal。
2. ALU 除了拿經過 sign extend 的 immediate 外為了做 addiu 等 unsigned 的運算所以也傳入了原始的 immediate
3. Load 和 store instruction 是要存取多少 byte 我留到 Memory 在做判斷所以在 memory 的部份也傳入了 op code
4. Memory multiplexer 除了原本判斷傳回 memorydata 還是 ALU result 外，我將 write back to register 的部份也一併處理，所以 RegWrite 訊號改為傳到這。因為 JAL 也會改變 Register 的值所以 JAL_sig 訊號也傳入這。
5. PC control 負責處理 branch、jump、jr 是否發生的不同狀況
6. 我將 Error detect 扣除 pc 的 nuber overflow 外統一在一起 detect，write to $0 的部份需要注意的是 NOP，以及 sll $0 $0 0 即使 rs 不為 0 亦為

NOP。Number overflow 要注意的地方是負數最大值的減法以及 unsigned add 的部份。misalignment 要注意的是即使 offset 小於 1023 但若是存取 BYTE 或 HALF BYTE 時也有可能會超過 1023

7. PC overflow 因為會等到下個 cycle 使用 pc 時才會發生所以 detect 的時間和其他狀況分開
8. Pc 跑到初始值之前時執行 NOP

2) Test case Design

2-1) Detail Description of Test case

DMemory 的設計想法:

1. 考慮到負數最大值在許多運算以及 error detect 中會有較多需要處理的狀況(ex: 正-(負數 max) or 負-(負數 max)的 number overflow 狀況)所以放入負數最大值
2. 不需做 write to register 的狀況只有 Sll $0 $0 0,所以我放 0x00 在 DMemory 中之後將 0x00 load 到$17 再做 instruction sll$0 $17 0 , 檢查是否有錯誤理解 NOP 的狀況
3. 設計一個 0x00 及 0x03 的值,在後續設計 BNE 指令讓 PC 跳到起始 PC 值以前時，可以在重複跑到第三次時不執行 BNE 指令，往下一行 instruction 執行
4. 其他為隨意的正負值參雜，方便之後設計 Number Overflow 狀況時使用

IMemory 的設計想法:

1. 設計 PC 跑到 PC 初始值以前的狀況，檢查是否會正確執行 NOP 指令
2. 設計 lw $5 1($4)，考慮 immediate 不是 4 的倍數但 ALU result 結果為對齊的狀況
3. 設計負數最大值的 SUB，確認是否有考慮到負數最大值的 Number Overflow 狀況
4. 因為 ORI 的 immediate 不需要做 sign extension 所以加入 ori $8 $1 FFFF 確認是否有考慮到 ORI 的 unsigned C
5. 設計 addu 的 number overflow 但不需進行 error detect
6. 設計 beq 指令在 ALU 進行減法時產生的 number overflow，但不需 detect error
7. 加入 write to register$0 跟 number overflow 的 error
8. 所有指令都簡單執行一遍，確認所有指令都正確指行