112學年度第一學期計算機組織
期末專題
Pipelined MIPS CPU Simulator
專案報告

第四組
A1105513 莊印哲
A1105522 劉川愷
A1105535 鄭芃予

指導教授:
郭錦福 教授

1. 概要

本次專題旨在製作pipelined的cpu模擬器，盡可能使程式執行方式貼近實際運作情形。

2. 主架構

主架構分為兩部分，資料結構和函式。為了使程式更加貼近實際運作情形，我們透過設計資料結構，利用資料結構包裝在Stage間傳遞的資料。並以函式的方式區分各Stage區塊，以模擬各Stage所執行的內容。

2.1. 資料結構設計

- Instruction(Class)
  用於儲存Instruction的名稱和下一個cycle的stage
- stageInstruction(Dict)
  以各個stage名稱為key，儲存各個stage的Instruction
- pipInfo(Class)
  用於儲存各Stage 的 pipeline registers
- PipelineRegister(Class)
  以四個Dict 分別儲存IF/ID、ID/EX、EX/MEM、MEM/WB
  並以key分成input和output

2.2. 函式

- IF、ID、EX、MEM、WB
  模擬各stage的工作
- check_beq_hazard
  檢查beq 指令是否需要forwarding或stall
- check_ex_hazard
  檢查EX stage是否需要forwarding或stall
- check_mem_hazard
  檢查是否需要forwarding

2.3. 流程設計

1. 執行各Stage函式
2. 決定下一輪IF讀取的指令(beq、stall判斷)
3. 將指令移動到下一個Stage
4. 將pipInfo從output移動到input

3. Stage

3.1. IF

- 變數currentInstructionNum作為PC。
- 根據原本指令順序或beq加減來fetch指令
- 根據以上設定IF/ID暫存，但只有name與instuction本身，其餘交於ID做解碼。

3.2. ID

- 接收IF/ID output對指令進行字串切割、解碼，包括解析指令的rs,rt,rd,offset等Operand。
- 根據Operand判定是否發生hazard，並設定flag(是否stall或forwarding)以便後續處理。
- 若指令為beq便於此判定beq hazard，根據判定來進行fowarding，隨後判定是否要jump。
- fowarding是先獲取ID/EX的暫存(Pipeline Register)。
- 根據以上設定ID/EX暫存。

3.3. EX

- 根據flag判斷是否進行forwarding，若判斷為True，則於此進行fowarding以利計算。
- fowarding是先獲取EX/MEM的暫存(Pipeline Register)。
- 若指令為R-format，則計算運算結果，並存入暫存(EX_result)。
- 若指令為I-format，則計算好所要獲得的memory位址，並存入暫存(EX_result)。
- 根據以上設定EX/MEM暫存。

3.4. MEM

判定指令類別，並根據EX運算結果執行動作。

- 若是sw:將指定memory設為指定register的值。
- 若是lw:將指定要存回register的值存放至MEM/WB暫存(loadword)。
- 根據以上設定MEM/WB暫存。

3.5. WB

判定指令類別，並根據MEM執行結果動作。

- 若是sw:不會做任何事情，因為sw會在MEM將register的value寫到memory中

- 若是lw:將MEM讀取的資料(loadword)寫到指定的register中。
- 若是R-format:將指定register設為指定為MEM所傳遞的值(ALU運算結果)
- 若是beq : 則不做任何事情

4. Hazard

利用PipelineRegister(各部件暫存)及傳入參數(ID解碼後的rs,rt)來判斷是否發生hazard並設定flag以便後續處理。

4.1. EX_hazard

- 首先根據PipelineRegister判定EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)條件。
- 若是R-format指令且rd與rs或rt相同，設好flag以讓之後物件進行forwarding。
- 若是I-format指令且rd與rs或rt相同，進行stall設定(將ID的stageInstructions設成自己)。

4.2. MEM_hazard

- EX_hazard為先決條件。否則不進行MEM_hazard判定。
- 根據PipelineRegister判定MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)條件
- 若是R-format或I-format指令進行forwarding。

4.3. beq_hazard

- 判斷前前一個指令跟前一個指令均為lw，且beq的rs或rt跟前一個lw的rt位置相同，需要做兩次stall。並用flag存起來，在下一次Cycle的ID,IF階段直接做stall
- 判斷前一個指令為R-type，且beq的rs或rt跟前一個R-type的rd位置相同，則做一次stall
- 每次做stall時候，都會把當前Cycle記下，做驗證
- 判斷前一個指令為R-type，需要確定該rd有船進MEM，需要做forwarding

5. 困難點

5.1. Stall做法

原先Stall的做法是使指令停留在原先的Stage將指令重做一次，但在beq撰寫時遇到問題。由於會影響到IF所讀取的指令，因此需要等到Stall完後再執行。

- 解決方式：

將Stall提前到Stage開始時判斷，若需要Stall則不執行。

5.2. Pipeline 跟 StageInstruction 指令不同步

原本PipelineRegister只使用單邊暫存而已，但這樣導致了Stage所抓的指令會提前抓到下一個指令。因此在Stage所讀到的指令會不一樣，也導致了會做錯功能。

- **解決方式：**

  將Pipeline分割成input/output，依照MIPS原本內部架構去做更改，並且也可以對後續的forwarding跟stall做更好的取值跟處理

5.3. PC抓取指令問題：

我們原先在計算下一個fetch的指令時是由各部件對PC變數做加減，以達到jump或stall的效果，但這樣的處理方式可讀性差，易造成程序出錯。

- **解決方式：**

  只由主程序及beq對PC變數做變動，以防意外更動。

6. 心得

莊印哲

在這次的專題製作過程中，我對 MIPS 的內部架構和運作原理有了更深入的理解。特別是對於 Pipeline 的設計，我明白了為何需要將其分為 input 和 output 兩部分。如果只設計成單邊暫存，每次 Cycle 的指令都會提前執行(偷跑)，這是我們一開始遇到的最大問題。除了 Pipeline 的設計，我也對 MIPS 指令在每個Stage在做甚麼事情比較了解。再來是程式整合重要性，當初設計的時候每個仁程式都是分開的，要merge的時候差太多，反而沒辦法好好合併，希望透過這次專題能理解程式整合的重要性。都要透過實際coding和執行指令，我學習到了如何有效地使用不同指令在各種不同的Stage來完成特定的任務。這次的專題製作讓我明白了理論知識和實際操作之間的差異。雖然我在課堂上學習了 MIPS 的理論知識，但在實際操作中，我還是遇到了許多我從未想過的問題。我會覺得理論知識和實際操作相結合的重要性，我覺得這可以反映到各個專題，尤其是畢業專題等，因為大家通常都是先去理解內部原理在做甚麼很少會直接去實作。反而都是要做的時候才發現有一堆問題存在。

劉川愷

在這次專案中，我更深入的了解了計算機組織以及MIPS結構，透過對MIPS的實現能更家理解各指令的運作過程。其中最重要的

是體驗到了Pipeline的設計流程，為了用最高的效率完成動作，各部件的分工是相當明確且嚴謹的，缺少一個步驟或多一個指令都會造成bug，因此理清實際原理與我們自記設計的架構非常重要，設計時也必須時時刻刻注意各部件是否發生衝突，像是在前面設定stall可能會造成後面部件的停擺、flag設定可能會造成其他條件判斷失誤。

在完成這次專案的過程中，我們也感受到團隊間討論的重要性，如果沒有做好溝通或程式的加註會造成coding時間成本增加。良好且有效率的分工也是必須的。

鄭芃予

在這次專題中讓我更加了解Pipelined的運作架構，也在實作過程中體會到專案管理的重要性。最一開始因為每有統合大家的程式架構，導致每個人對於程式撰寫的想法都不太一樣，在一次討論後我們統整了大家的想法，確認了程式架構，雖然需要修改已撰寫的程式，但可以讓後續開發更加順利。在這次的專題中我負責的是主架構程式、資料結構設計、程式碼整合和 Stage IF、ID。在撰寫beq判斷及最後整合時，發現了不少問題，雖然程式是可以運作的，但再加上beq的判斷就會有許多問題浮現出來。像是beq和stall都會需要控制IF所讀取的指令，在尚未整合時每個部分都會各自操作IF所要讀取指令的變數，導致兩部分會互相干涉，因此最後將決定IF下一個指令的部分改到Cycle結束後一次判斷。另外還有遇到大家對於變數的理解不一致，又由於Python弱型別的特性，導致程式出錯難以發現。幸運地在大家的努力下錯誤都成功處理完了。這次專題讓我深刻體會到團隊溝通及專案管理的重要性，可以節省大量理解他人程式碼與整合上的時間，大大加快專題製作速度。

7. 小組分工

姓名	莊印哲	劉川愷	鄭芃予
工作內容	架構討論		
	Stage : EX、WB Hazard : EX、BEQ stall 設計、pipeline	Stage：MEM Hazard：EX、MEM forwarding設計	Stage:IF、ID 主架構程式 資料結構設計 程式碼整合