普級 CPU 設計與製作

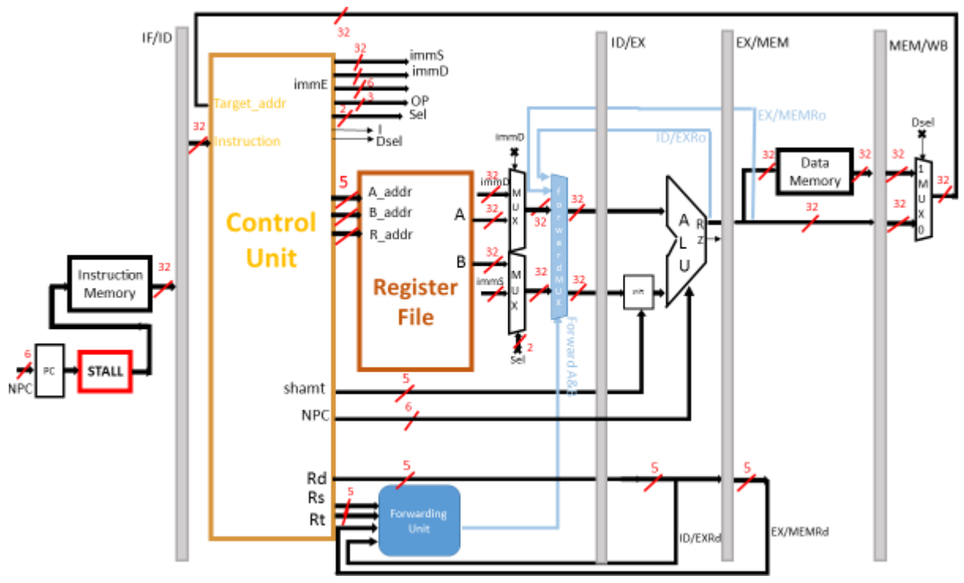
參展人員：尤寧、鄒廷東

一、動機

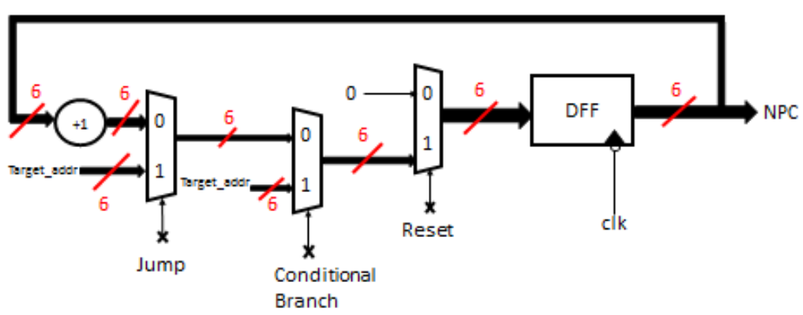
市面上的中央處理器大多具備多功能的運算能力，而導致一顆中央處理器要價不菲，本專題希望將中央處理器運用在日常生活的各項用品，並有簡易的資料運算功能，我們簡化傳統中央處理器來降低成本，同時也具備多種功能。

二、CPU 架構

本專題以 MIPS 為基礎，做結構上的簡化、指令重新設計以及部分模組的創新。我們將中央處理器做管線化切割成五個階段：指令擷取 IF、指令解碼 ID、執行 EXE、資料記憶體讀寫 MEM、寫回暫存器 WB 五階段，並且平均分布每個階段的關鍵延遲。主要可將整體架構(圖一)分成以下三個部分：



圖一整體架構圖



圖二 Program Counter

A. Control Unit

控制處理器訊號線，一旦處理器沒有控制單元，處理器將無法正常運作。

控制單元接收從 Instruction Memory 32 bits 的指令後，會在 Pipeline 第二階段將這 32 bits 做解碼，產生訊號線傳遞給之後的單元使用。這些訊號將控制整個處理器的運作方式。

B. **Processing Unit**

處理單元內有存放運算值和位址值的 Register File、Data Memory、Forwarding Unit、Mux、Shift Register，以及執行運算的 ALU。這些單元接受來自 Control Unit 解碼的訊號，再利用多工器選擇要將何種訊號丟入 ALU 執行運算，最後 ALU 運算完的結果會依指令的指示寫回暫存器或是寫入記憶體。

其中**前瞻單元**的設計是為了預防資料屏障 data hazard 的發生：在一般情況下當我們從 Register File 中某暫存器取值，該暫存器如果是上一條指令要存回的目標暫存器，取值的同時上一條指令並未執行到 WB 的階段，就發生資料屏障。所以我們設計此前瞻單元，放置在第二階段，將 ALU 運算的結果分別從第三階段以及第四階段送到此單元，供 A 和 B 的多工器選擇是否取用；第三階段拉回的 R 是因應上一條指令的狀態，第四階段則是因應上一條指令也用到 forwarding 的狀況。

C. **Program Counter**(如圖二)

存放下一個將要被執行指令所在的位置。在一般情況下 PC 每個 clock 會上數增加 1，並且在正緣觸發時將此位址傳給 Instruction Memory；當遇到特殊指令，例如跳躍和分支指令，PC 會等待新的位址被運算完再行傳送。

在 PC 更新之後，如果上一條指令是跳躍或是分支指令，該 PC 會被栓鎖在 Stall Unit 中，等待分支位址運算完之後才送出分支位址。

三、CPU 指令及功能介紹

Bit	31	30	29	28	27	26~24	23~19	18~14	13~9	8~0
Inst	sh	Br	M	I	W	op	Addr_R	Addr_A	Addr_B	Immediate

Instruction Table

為 32 bits 的 CPU，指令中包含 SH(位移單位)、Br(跳躍及分支發生)、M(寫入記憶體)、I(具有立即值)、W(寫回暫存器)、OP Code、暫存器位址以及立即值和位移量。

目前基本的指令包含 MOV、ADD、SUB、AND、ORR，以及 JUMP 和 BRANCH，其中 R-type 的指令皆分成兩種形式，一種是兩個來源暫存器作運算，另一種則是一個來源暫存器和立即值運算；另外，R-type 指令皆有位移量和位移方向的設計。

四、本專題實現之改良

1. 跳躍指令只浪費一個 clock-cycle

跳躍指令不經 ALU 運算，直接在第二階段的 Control Unit 發現該指令是跳躍後，直接將即將跳躍的絕對位址送回 PC；亦即 JUMP 可以在兩個 clock-cycle 內完成，相較於傳統的 MIPS，本專題直接省去一個階段的執行時間。

2. Stall Unit 設計防止分支指令取錯值

跳躍和分支指令會透過 Stall 來栓鎖 PC，跳躍指令鎖一個 clock-cycle，分支則是所兩個，目的在於等待這些指令更新跳躍即分支位址。

3. Forwarding Unit 提前運作

為了平均每個階段的 gate delay，前瞻元件放置在第二階段以供提前判斷。

五、未來展望

在未來的研究方面，本專題將以人為出發點，規劃最佳能普及化的 CPU。主要重點在於減少每個指令所需要耗費的時間，藉此加快 CPU 運作的速度；我們會從雙執行續並行處理指令開始，利用低於兩倍的處理器空間達到兩倍以上的速度增進。