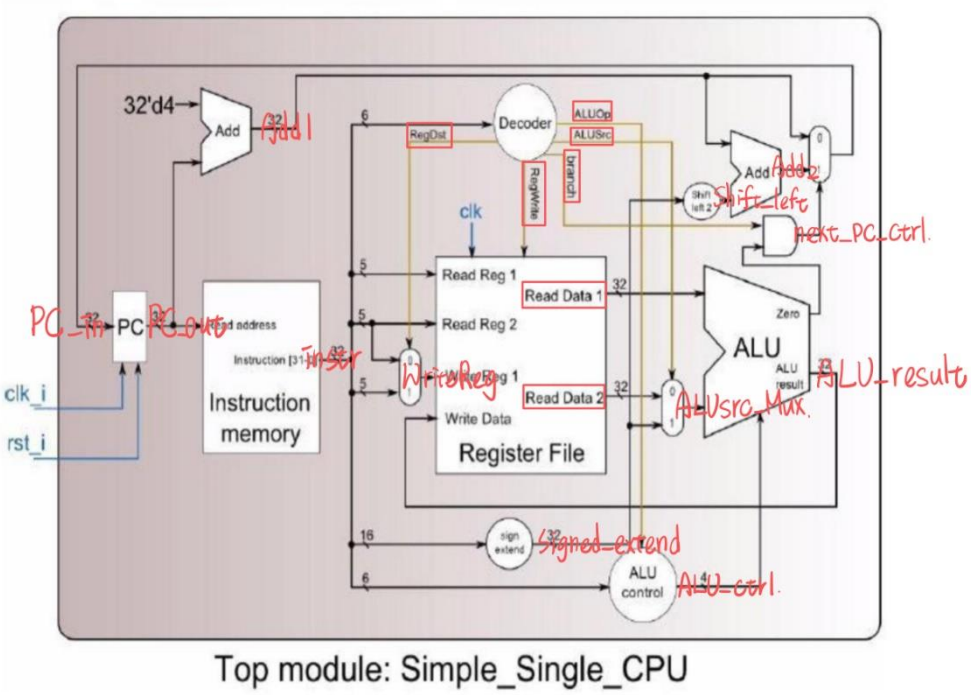
Computer Organization Lab2

Name: 孫宜君

ID:109550118

Architecture diagrams:



上圖紅色標記為 Simple_Single_CPU.v 中每條線路的命名。

LAB2 實作的部分包含 ALU 運算單元、Adder 加法器、MUX 2*1 選擇器、Shift Left Two 位移器及 Sign Extender 有號擴充器，將 funct_i 及 ALUop_i 兩個 input 處理成 output 訊號 ALUCtrl_o 輸出，經由 Decoder 將收到的 instr 根據講義解碼成五個訊號給以 wire 連接 CPU 內部的 13 個子模組使用。

Hardware module analysis:

使用多層次的方式來設計控制單元。

最上層是 1 個 Simple_Single_CPU( ) module，為了維持架構不做實際運算，僅連接各個 module。

主要 CPU 包含 3 個 MUX 選擇器、2 個 Adder 加法器、1 個 ALU 算術單元、1 個 ALU_Ctrl 控制訊號、1 個 Decoder 解碼器、1 個 Instr_Memory 記憶體、1 個 ProgramCounter 計數器、1 個 Reg_File 暫存器、1 個 Shift_Left_Two_32 位移器、

1 個 Sign_Extend 有號擴充器。

(1) Program Counter:

存放下一個要被執行指令所在的記憶體位址。

(2) Adder:

有別於 ALU，此加法器用於

1. 增加 PC 值已只到下一個指令的位址
2. 計算 branch 指令的 target address

(3) Instr_Memory:

讀進 PC 指定所要讀取指令所在的記憶體位址，輸出那個 address 的指令。

(4) Decoder:

讀進 Opcode，輸出對應的 control。

(5) MUX:

依條件選擇對應的輸出。

(6) Sign Extend:

將 16bits 變成 32bits

(7) ALU Control:

根據 Decoder 傳出的 ALUop 來決定 ALU 要做哪個運算

Pro: 使用小的控制單元來達到多層次的控制可以降低主要控制單元尺寸及加快控制單元的速度。

(8) ALU:

將兩個 input 的數值根據 ALU control 的值做不同的運算

(9) Shift Left:

將輸入向左平移 2 位。

(10) Register File:

是一群暫存器的集合，透過暫存器編號，可 Read 或 Write Register File 裡的任何一個 Register。

Finished part:

Testdata1

```
r0=          0
r1=         10
r2=          4
r3=          0
r4=          0
r5=          6
r6=          0
r7=          0
r8=          0
r9=          0
r10=         0
r11=         0
r12=         0
```

Testdata2

```
r0=          0
r1=          1
r2=          0
r3=          0
r4=          0
r5=          0
r6=          0
r7=         14
r8=          0
r9=         15
r10=         0
r11=         0
r12=         0
```

Problems you met and solutions:

1. 一開始不懂為甚麼 Decoder 是將 3bits 的指令當作 ALUop 傳給 ALU_ctrl，後來回去翻講義才知道正確的運作原理
2. Vivado 還是和 lab1 的時候一樣會無法預期的突然死掉整個卡住，但是後來發現按照助教在臉書社團建議的，每次都在 TCL console 打 close_sim 以確保正確關閉，就有效的減少了卡死的狀況，但偶爾還是會發生。

Summary:

透過這次 lab 實作 CPU 內的架構，實作的過程中更了解了每個元件的運作原理及訊號控制，verilog 的運作已經逐漸上手了，在這次 lab 實作過程中更注意 reg、wire 和 assign 的使用，避免了很多 error 的產生。