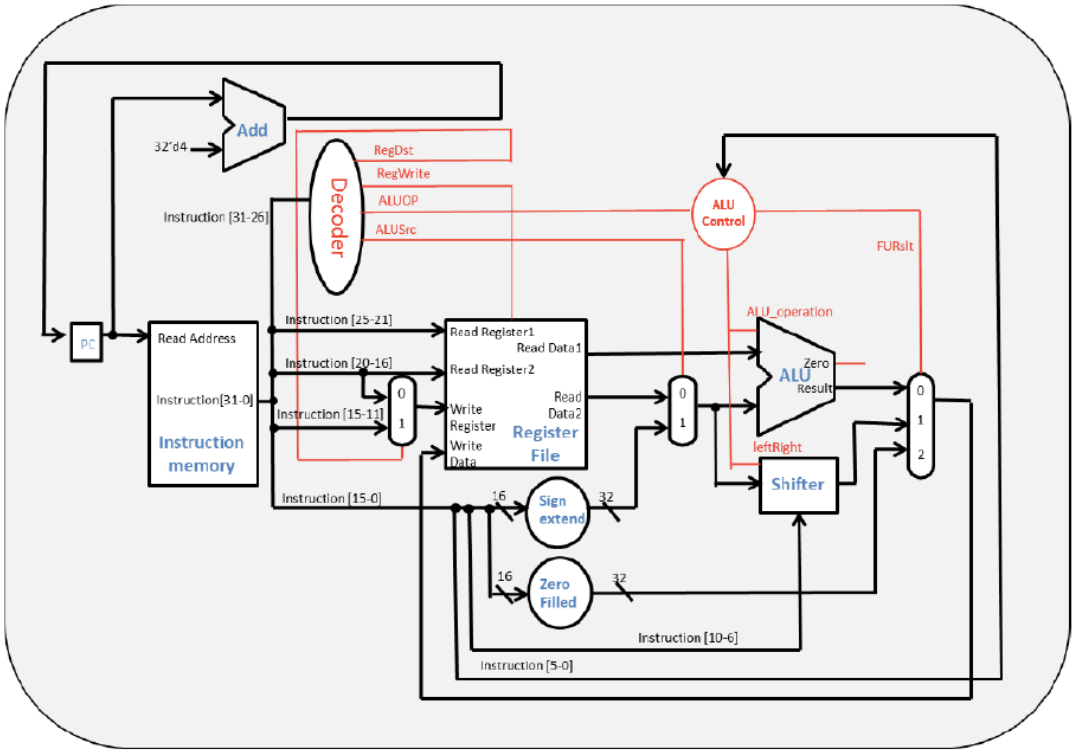
Computer Organization

Architecture diagrams:



Hardware module analysis:

Program_Counter

將當前位址送往 Instruction Memory 以讀取指令。

Adder

用於將當前位址+4 來更新 PC 的加法器。

Instr_Memory

讀取當前位址在記憶體裡對應的指令。

Mux2to1

2 to 1 的 Mutliplexor。第一個是在 rt 與 rd 之間挑選，第二個是在 rt 與 sign extension 中做選擇。

Reg_File

讀取各個 Register 所儲存的資料，亦可用於將資料寫回 Register。

Decoder
根據指令分配各個 Multiplexor 與 ALU 運行時所需的 control signal。

ALU_Ctrl
根據不同的 signal 設定 ALU operation 以決定 ALU 該執行什麼行為。

Sign_Extend
將 16-bit 的數值做 sign extension 到 32-bit。也就是將 16~31 位的 bits 都複製為第 15 位的 bit。

Zero_Filled
將 16-bit 的數值做 zero filling 到 32-bit。也就是將 16~31 位的 bits 都設為 0。

ALU:
根據 ALU operation 以執行 ADD、SUB、AND、OR、NOR、SLT、SLL、SRL、ADDI 等運算。

Shifter
根據 shamt 來對數值進行左移或右移。

Mux3to1
3 to 1 的 Mutliplexor。從 ALU、Shifter、Zero_Filled 中決定輸出的值。

Finished part:

全數要求的 verilog 檔案皆已完成。

Problems you met and solutions:

1. Simple_Single_CPU.v 需要手動補上的部分
沒有看討論區，自己 debug 很久都找不到問題，後來幸好還是有注意到需要自己接線，以後還是要多看討論區啊，至於因為眼花而接線接很久就是另一個故事了…
2. AND 與 OR 的 ALU function code 與講義上相反

透過測資上的報錯，檢查後發現 HW2 裡設計的 ALU 裡 AND 是 0001，OR 是 0000。與上課講義正好相反，經過修正後就正常了。

3. SLL 結果是錯的

後來發現因為使用 Appendix 提供的 Shifter.v，裡面的左右判斷與我設計的 ALUOP[0]正好相反，把 Shifter.v 判斷式裡顛倒就解決了，以後還是要先事前確認。

Summary:

本次作業實作一個功能有限的 single-cycle CPU，整體難度看似很高，但拆成各個零件分別處理以後難度也下降許多，不過 debug 還是有些複雜，要根據報錯的功能去思考其在 CPU 裡執行的路徑，依序檢查使用到的零件是否出現問題，略為繁瑣。但是寫完以後充滿成就感且更熟悉 CPU 運作。