Computer Organization Lab2 CPU

0816146 韋詠祥

Architecture diagrams:

原先提供的架構已包含 Instant Memory、Program Counter、Register File 這三個子模組的完整實作

這次作業首先利用 Verilog 撰寫 ALU 運算單元、Adder 加法器、2x1 MUX 選擇器、Shift Left Two 位移器、Sign Extender 有號擴充器這五個基礎元件

再依照 spec 及講義內容，完成 ALU Control 控制訊號，將 `funct_i` 及 `ALUOp_i` 兩個輸入處理轉為 `ALUCtrl_o` 輸出

最後根據 spec 的 Architecture Diagram 將 CPU 內部 13 個子模組用 `wire` 連接起來，並將 Decoder 解碼器收到的 `instr_op_i` 根據講義解碼成五個訊號給其他子模組使用

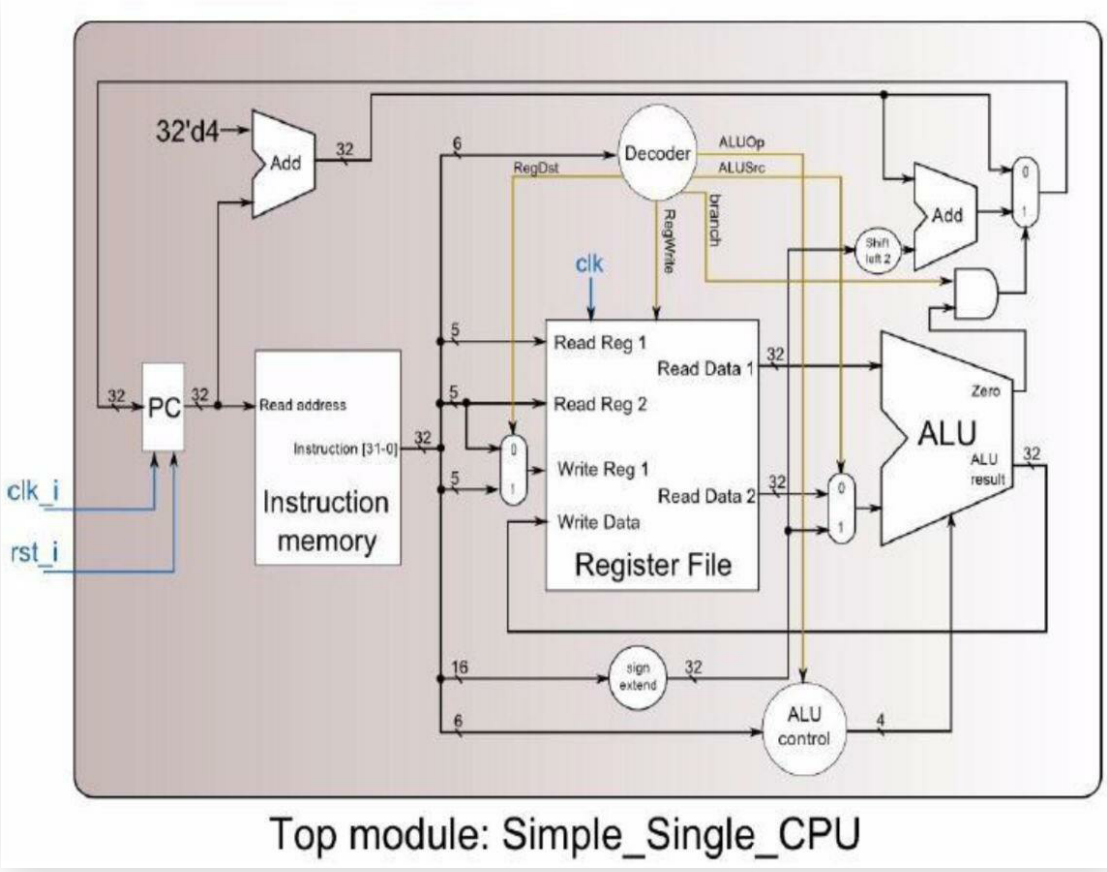

Fig 1. 架構圖，擷取自作業規格

Hardware module analysis:

最上層是 1 個 Simple_Single_CPU( ) 模組，僅連接各個子模組，為了維持架構簡潔不做實際運算

主要 CPU 模組內裡面包含 3 個 MUX 選擇器、2 個 Adder 加法器、1 個 ALU 算數單元、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 有號數延伸器，每個子模組都只有 20 - 50 行的 Verilog 程式碼

Finished part:

完成所有模組後，成功讓兩個測試資料的輸出與 spec 上的答案相符

```
[3:23:47] sean :: Sean-MBPR  ➜  Repos/NCTU-109B-Comp-Org/Lab2-CPU <master*> » make
cp testcases{1,}.txt
perl -pe 's/(reg \[32\-1:0\] Instr_Mem \[0:)\d+\-1\];/${1}'"$(cat testcases.txt | wc -l | tr -d ' ')"'-1];/' -i Instr_Memory.v
iverilog -o cpu ALU.v ALU_Ctrl.v Adder.v Decoder.v Instr_Memory.v \
                MUX_2to1.v ProgramCounter.v Reg_File.v Shift_Left_Two_32.v \
                Sign_Extend.v Simple_Single_CPU.v Test_Bench.v
./cpu
VCD info: dumpfile debug.vcd opened for output.
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
cp testcases{2,}.txt
perl -pe 's/(reg \[32\-1:0\] Instr_Mem \[0:)\d+\-1\];/${1}'"$(cat testcases.txt | wc -l | tr -d ' ')"'-1];/' -i Instr_Memory.v
iverilog -o cpu ALU.v ALU_Ctrl.v Adder.v Decoder.v Instr_Memory.v \
                MUX_2to1.v ProgramCounter.v Reg_File.v Shift_Left_Two_32.v \
                Sign_Extend.v Simple_Single_CPU.v Test_Bench.v
./cpu
VCD info: dumpfile debug.vcd opened for output.
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
```

Fig 2. 實驗結果

Problems you met and solutions:

1. 剛開始的時候對 MIPS 指令格式還不夠熟悉

像是 `Decoder` 要將哪 3 bit 的指令當作 `ALUOp` 傳給 `ALU_Ctrl`，花了一段時間重新讀講義才理解該怎麼做

2. 不肯定自己是否有正確實作

雖然兩個測試資料都有得到正確的結果，不過由於這次沒有自己產生測資，在最佳化寫法時很怕不小心就改壞東西

3. 報告不會寫

程式幾小時就寫完了，但 report 一直生不出來 QwQ

可以的話希望以後能提供參考範例，讓同學們知道該往哪些方向寫，助教們改起來格式統一也比較輕鬆

Summary:

經過了這次的 Lab，我更了解 CPU 運作的原理，也藉由實作過程中釐清了自己對 MIPS 指令格式的疑惑，讓計算機組織不再是背科

相較之前的我，現在或許更能靈活運用所學知識，為接下來的課程內容做準備