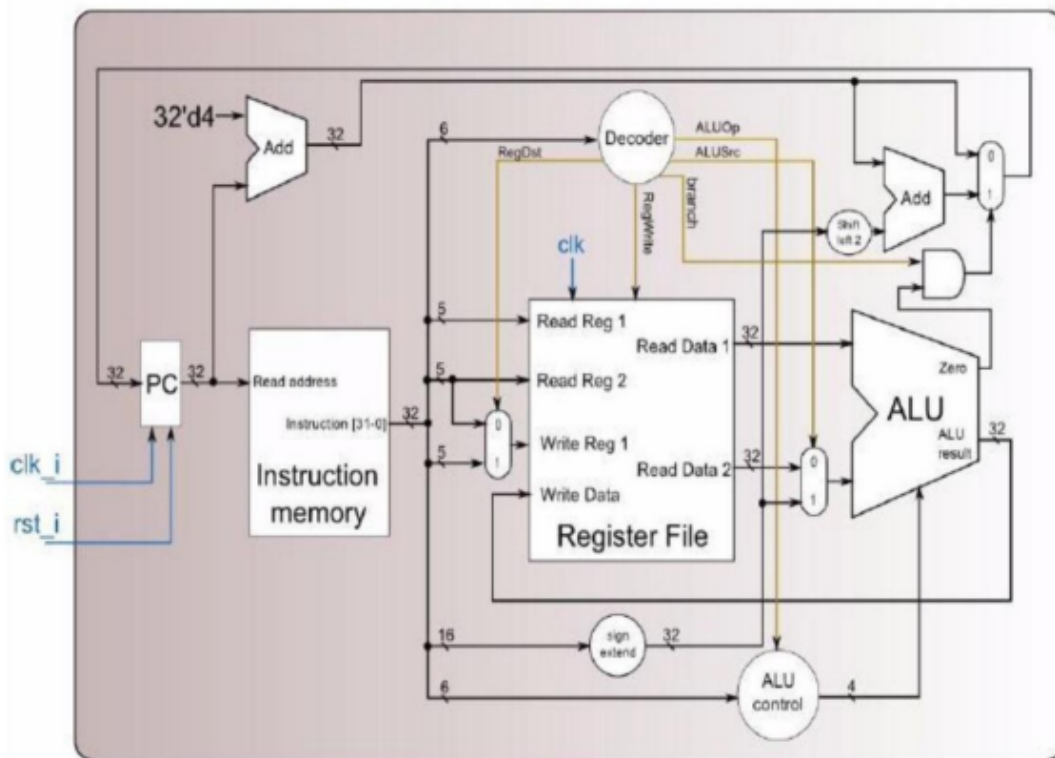


Computer Organization Lab2

Name : 郭昀 ID : 109550018

Architecture diagrams:



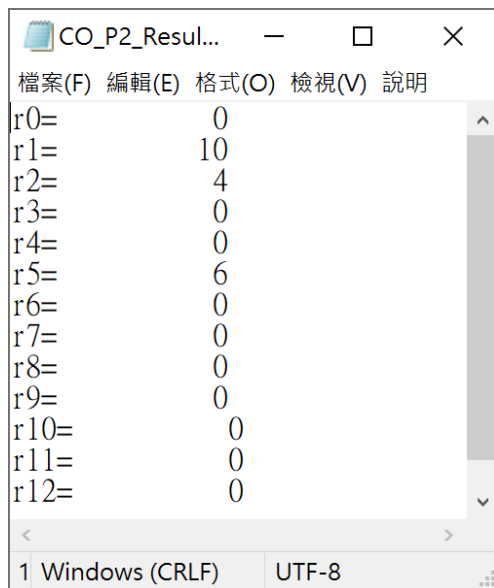
Top module: Simple_Single_CPU

Hardware module analysis :

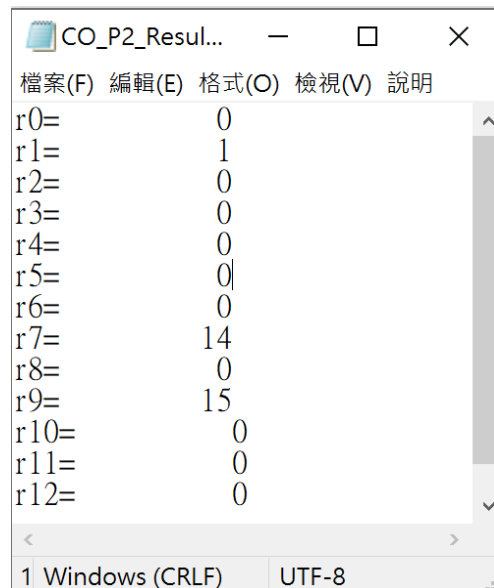
主要都是按照 Spec 裡面的 diagram 來設計我的 Simple_Single_CPU, 例如 :

- PC 透過左上的 adder 每次加 4, 而下一次的 PC 會在右上的 MUX 中由 ALU zero 決定是加 4 的那一個, 還是右上的 adder 算出來的 branch target。
- Register File 根據 Instruction memory 傳來的是 R-format 還是 I-format 決定要從哪裡讀值。
- Decoder 是用來判斷 instruction 所需對應的 RegDst, RegWrite, branch, ALUOp, ALUSrc 的值。
- 如果 instruction 是 I-format, 就需要 Sign extender 把 16 bit 轉成 32 bit。
- ALU control 會決定 ALU 要做什麼運算。

Finished part:



```
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
```



```
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:

我一開始在沒有注意到在 ALU.v 裡面, zero 的輸出的名字是 zero_o 而我打成 zero, 導致我的 beq 指令都是錯的, 後來 debug 許久後才找到這個錯誤。

Summary:

做完這次的 Simple Single CPU, 讓我對 MIPS 的 Instruction 有更深入的了解, 更清楚他是如何處理這些指令, 以及這些指令相對應的輸出。