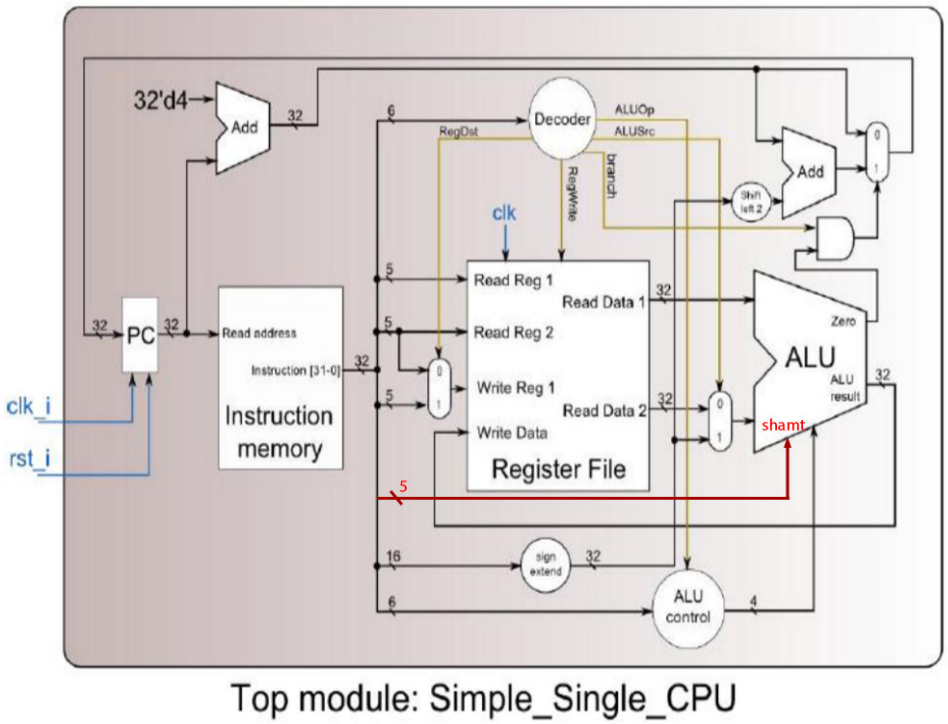
Computer Organization

Architecture diagram:



Top module: Simple_Single_CPU

Detailed description of the implementation:

指令	Opcode	ALU Op	ALU Src	Reg Dst	Branch	ALUCtrl	ALU 操作對應 Verilog Code
addu	000_000	0	0	1	0	2	`a + b`
subu						6	`a - b`
and						0	`a & b`
or						1	`a | b`
slt						7	`a < b ? 1 : 0`
sra						3	`$signed(b) >>> shamt`
srav						4	`$signed(b) >>> a`
addi	001_000	1	1	0	0	2	`a + b`
sltiu	001_011	2	1	0	0	7	`a < b ? 1 : 0`
beq	000_100	3	0	0	1	6	`a - b`
lui	001_111	4	1	0	0	5	`b << 16`
ori	001_101	5	1	0	1	1	`a | b`
bne	000_101	6	0	0	1	8	`a == b ? 1 : 0`

上表為各指令對應之 Control Signal

1. addu、subu、and、or、slt、srav：
   讀取 Rs、Rt 位置的暫存器資料，經過 ALU 進行對應的運算後，寫回 Rd 位置的暫存器。
2. addi、ori：
   讀取 Rs 位置的暫存器資料，並將最後 16 bit extend sign 之後，兩者經過 ALU 進行對應的運算，寫回 Rt 位置的暫存器。
3. lui：
   將最後 16 extend sign 之後，經過 ALU 左移 16 位元之後（最低 16 為為 0），寫回 Rt 位置的暫存器。
4. sltiu
   讀取 Rt 位置的暫存器資料，並將最後 16 bit extend sign 之後，根據兩者大小關係將 Rt 位置的暫存器，寫入 1 或 0。
5. srav：
   將 shamt 數值接到 ALU，讀取 Rt 位置的暫存器資料，將其資料右移 shamt 之後寫回 Rt 位置的暫存器。
6. beq：
   讀取 Rs、Rt 位置的暫存器資料，經過 ALU 相減，如果兩個相等，則 ALU 的 Zero 輸出為 1，在電路右上選擇經過 adder 加完後的 PC 存回 PC 暫存器。
7. bne：
   讀取 Rs、Rt 位置的暫存器資料，經過 ALU 判斷相等與佛否，如果兩個不相等，則 ALU 的 Zero 輸出為 1，在電路右上選擇經過 adder 加完後的 PC 存回 PC 暫存器。

Problems encountered and solutions:

實作 srav 時問題比較大，因為 srav 感覺上是類似 I-type 的操作，但卻是使用 R-type 的指令，最後選擇直接將 shamt 接到 ALU 上進行計算，不知道是不是好的做法。

Lesson learnt (if any):

做完這個作業後對簡單 CPU 有更清楚的了解，雖然因為沒什麼寫過 verilog 的經驗，一開始花了比較久的時間，不過還好最後能成功完成。