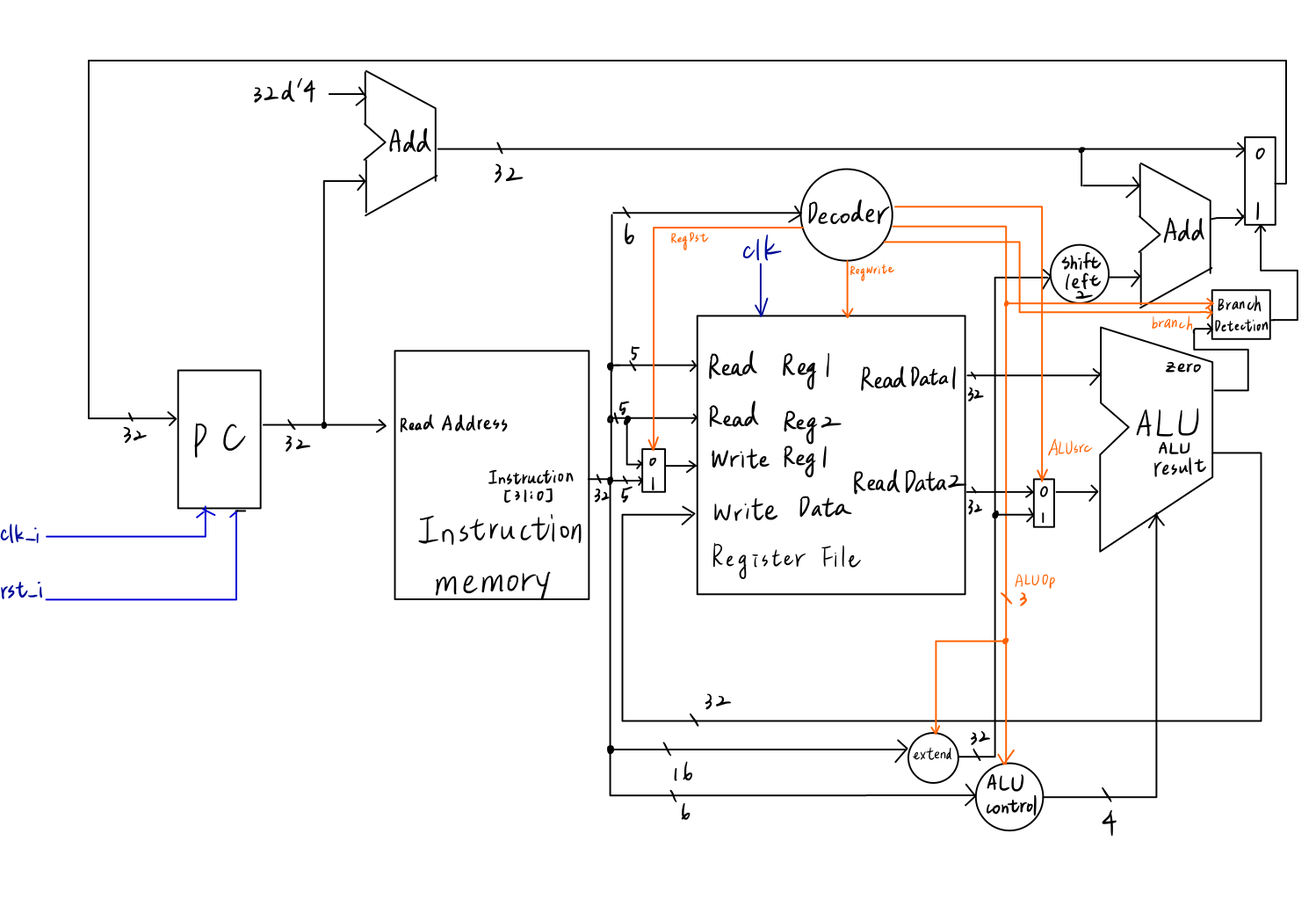
**Computer Organization**

**Architecture diagram:**



**Detailed description of the implementation:**

這次的Lab是要用上課所學到的用verilog實作一個簡單的single cycle cpu。

**Simple\_Single\_CPU**

主要是由外部的clk\_i與rst\_i來控制內部的運作，可以藉由目前的Program Counter來取得Instruction Memory，並經由Decoder發送控制訊號，經過一連串的選取後可以使得暫存器或是Program Counter內部的值改變。

內部主要由以下模組構成:

**Program Counter**

由clk、rst與pc\_in作為輸入決定pc\_o，每當clk正緣觸發時會將pc\_in傳給pc\_o。pc\_o存放的是Instruction Memory的地址。

**Instruction\_Memory**

輸入為目前要執行的指令地址，輸出則是32-bit的指令，而指令是根據地址從測資的txt檔所取得，一次會取得32-bit。

**Reg\_File**

內部存放32個暫存器，有四個輸入，兩個輸出，輸入有目前讀取的暫存器的地址\*2，與寫入的暫存器地址與資料，輸出則是讀取暫存器的資料\*2。

**Decoder**

根據指令中的Op code(instruction[31:26])進行解碼，並依據不同類型的指令輸出控制訊號給其他模組。

* RegWrite : 依據指令決定是否寫入Register\_File。
* ALU\_op: 可依據不同指令初步判定要讓ALU執行加法或是減法等等的運算，輸出3-bit的control給ALU\_Crtl。
* ALUsrc: 用來決定要執行運算的是Register或是Immediate。
* RegDst: 用來決定是要寫入的暫存器地址是rd(指令為R-type)還是 rt(指令為I-type)。
* Branch: 若指令為beq或bne則輸出1，其餘則輸出0。

**ALU\_crtl**

根據ALU\_op所輸出的訊號與指令中的Function field(instruction[5:0])並輸出控制訊號給ALU。

**Extend**

將immediate (instruction[15:0])的值進行extension，要注意的地方是只有ori是進行zero-extension其他的都是sign-extension，所以使用ALU\_op作為控制訊號。

**ALU**

將ALU\_crtl的訊號作為輸入，使ALU可以執行各種運算，並將結果送回Register File，zero output則是beq、bne會使用到。

**MUX**

總共有用到三個2-to-1MUX。

* Mux\_Write\_reg: 選擇要寫入的Register的地址是rt或是rd。
* Mux\_ALU\_src: 選擇作為ALU\_Src2的是要用rt或是immediate。
* Mux\_PC\_src: 選擇是要用PC+4或是immediate\*4+PC+4作為下一個的Program Counter，控制線則是架構圖中的Branch Detection，其內部接線為Branch && ((ALUOp == 3'b001 && ALU\_zero) | |(ALUOp == 3'b011 && ~ALU\_zero))，因為beq和bne取的值不同(ALU\_zero)，所以需要再看ALUOp是哪一個運算並對PC\_src做選擇。

**Adder**

用到兩個Adder。

* 計算PC+4。
* immediate\*4+PC+4。

**Shift\_Left\_Two\_32**

將immediate做extension的結果乘以四，beq或bne的指令可能會用到其結果。

**Problems encountered and solutions:**

**Ori :Zero extended**

原本是將所有的immediate都做sign-extension，後來才發現到ori這個指令是需要用zero-extension，但是原本的sign-extension並沒有control signal做控制，因此我們將Decoder解碼後的ALUOp作為控制訊號給extend，只有ori指令使用zero-extension，其餘則使用原本的sign-extension。

**Lesson learnt (if any):**

這次的Lab讓我們知道Single cycle ALU可以怎麼用verilog實作，也更熟悉mips的指令以及Instruction每一個區塊的意義，知道每一個模組需要執行什麼功能，並將它們接在一起。