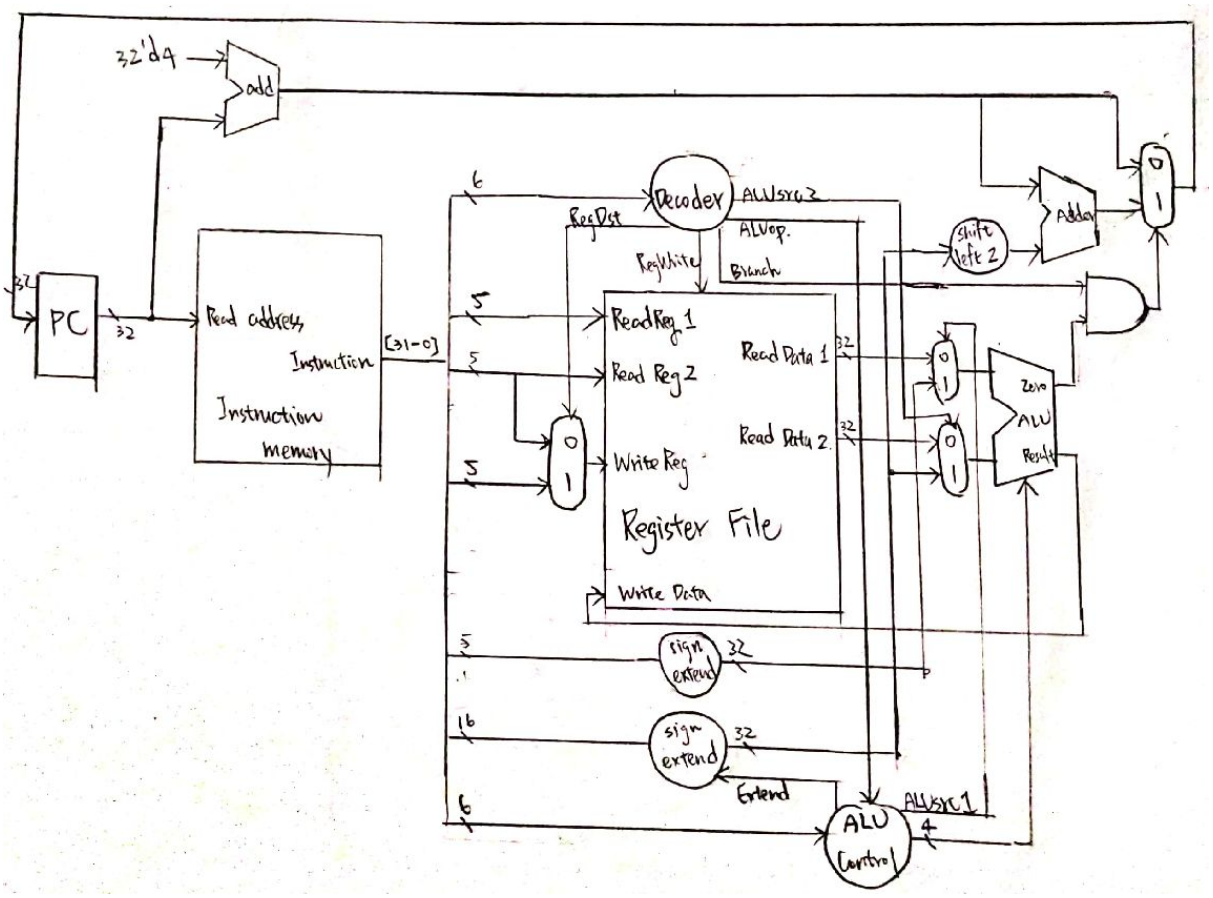
Computer Organization Lab-2

0616014楊政道, 0616225張承遠

Architecture diagram:



Detailed description of the implementation:

1. Simple_Single_CPU (列出更改spec圖的部份)

(1) shamt

由於spec中多了sra操作, 除了intruction要多拉出一個shamt的資料外，ALU中src1的值也需要多一個MUX來做兩個資料的區隔。當ALU要做shift的操作時，src1要接入shamt資料，src2則要接上被shift的資料，並且新增對應的control signal來操控他們。

(2) Signed_Extend_Ctrl

由於有些I-type操作有區分 signed 跟 unsigned，在sign extend的處理上也要根據操作上的不同來做出不一樣的sign extend。如果要用 signed 的方式來做sign extend，要補上16-bits數字中最高位的那個bit; 反之，要用unsigned的方式則全部補0即可。

2. Control Signal Table

Instruction	Opocde	Fuction	RegDst	RegWrite	branch	ALUSrc1	ALUSrc2	ALUOp	ALU_Ctrl	Extend
addu	000000	100001	1	1	0	0	0	000	0010	x
addi	001000	-	0	1	0	0	1	010	0010	1
subu	000000	100011	1	1	0	0	0	000	0110	x
and	000000	100100	1	1	0	0	0	000	0000	x
or	000000	100101	1	1	0	0	0	000	0001	x
slt	000000	101010	1	1	0	0	0	000	0111	x
sltiu	001011	-	0	1	0	0	1	111	0111	0
beq	000100	-	x	0	1	0	0	011	0011	1
sra	000000	000011	1	1	0	1	0	000	1000	x
srav	000000	000111	1	1	0	0	0	000	1001	x
lui	001111	-	0	1	0	0	1	101	0101	1
ori	001101	-	0	1	0	0	1	001	0001	1
bne	000101	-	x	0	1	0	0	100	0100	1

Problems encountered and solutions:

一開始看到有這麼多的source code感到有點頭痛，但跟隨著講義上內容及說明文件裡的架構圖走一遍之後，思路便開始清晰許多。再來愈到的困難便是找出`decoder`，`ALU_ctrl`及'ALU`三個檔案之間的聯繫，一個沒注意很可能就會導致嚴重的錯誤(位址亂跑，ALU執行錯誤的Function等等)，解決辦法沒有捷徑，只能根據波型圖的結果慢慢比對，經過幾次錯誤之後最後終於完成這次的Lab。

Lesson learnt (if any):

這次的Lab讓我對CPU的架構有了更深一層的認識，原本認識的CPU只僅限於表面而已，這次Lab做完後才發現CPU裡面的元件各個環環相扣，彼此互相影響。雖然這次的Lab只是模擬僅單的CPU而已，但也耗費了我不少時間，不過我相信這對於之後的課程將會有很大的幫助。

How to compile:

```
iverilog Test_Bench.v Simple_Single_CPU.v ProgramCounter.v Adder.v Instr_Memory.v
MUX_2to1.v Reg_File.v Decoder.v ALU_Ctrl.v ALU.v Shift_Left_Two_32.v Sign_Extend.v
```