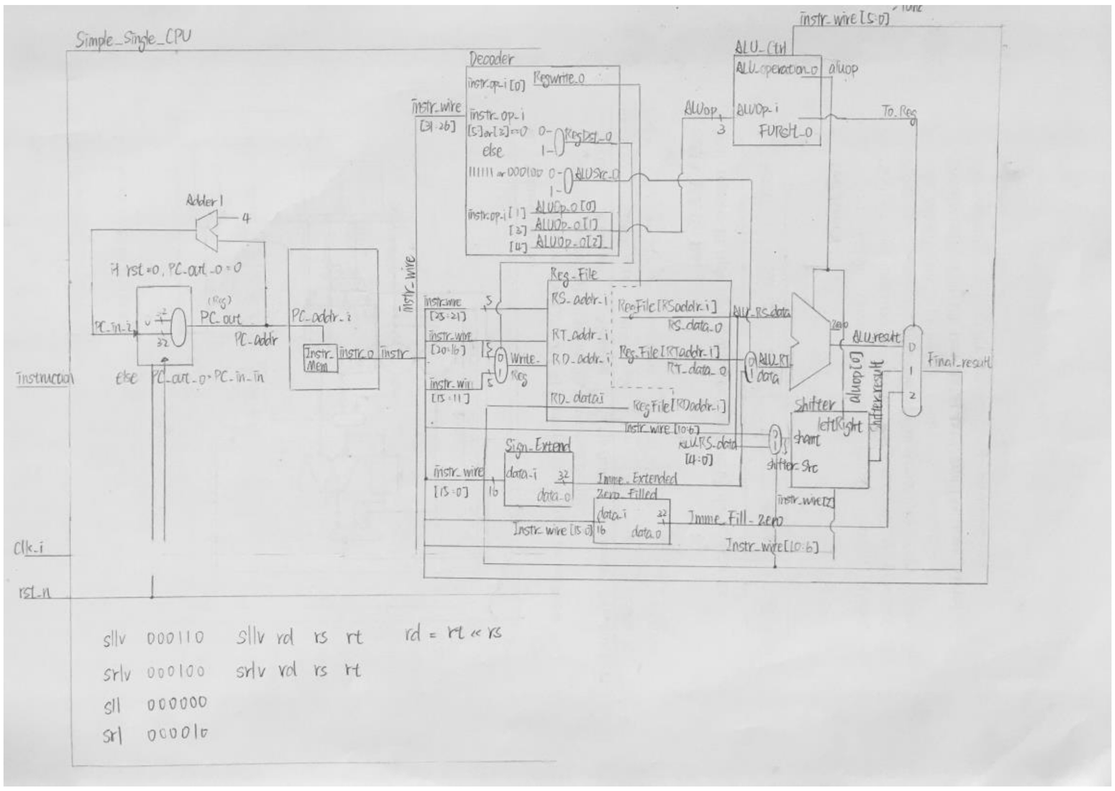
Computer Organization

Architecture diagrams:



Hardware module analysis:

Program_Counter PC:
PC_in = PC_out + 4
Rst=0 -> PC_out=0

Adder Adder1:
Out = In1 + In2;

Instr_Memory IM(32word):
Input:PC_Addr
Output: Memory 中位置 PC_Addr 的指令(32bit)

Mux2to1 Mux_Write_Reg:
由 select 變數決定輸入的兩個變數誰為最後輸出

Reg_File RF:

由 IM 輸出的 instruction RS RT RD 位置，取出 RS RT 值，若 wire RegWrite=1，將 input RD_data 寫進 RD register。

Decoder Decoder(instr_op_i, RegWrite_o, ALUOp_o, ALUSrc_o, RegDst_o):
由 Instruction OPcode[31:26] 接進 instr_op_i，
ALUOp_o[0]= instr_op_i[1]、
ALUOp_o[1]= instr_op_i[3]、
ALUOp_o[2]= instr_op_i[4]

若 instr_op_i=111111 or 000100(R-type or BEQ)，ALUSrc_o=0，else ALUSrc_o=1
若 instr_op_i[0](BEQ)，RegWrite_o=0，else RegWrite_o=1
若 instr_op_i[5] or instr_op_i[3] == 0，RegDst_o=0，else RegDst_o=1

ALU_Ctrl AC(funct_i, ALUOp_i, ALU_operation_o, FURslt_o)
若 ALUOp_i = 111 (R-type)
case(funct_i)
```
        FUNC_ADD : ALU_OP = 4'b0010;//2
        FUNC_SUB : ALU_OP = 4'b0110;//6
        FUNC_AND : ALU_OP = 4'b0000;//0
        FUNC_OR  : ALU_OP = 4'b0001;//1
        FUNC_SLT : ALU_OP = 4'b0111;//7
        FUNC_NOR : ALU_OP = 4'b1100;//12
          FUNC_SLLV: ALU_OP = 4'b1011;//11
          FUNC_SRLV: ALU_OP = 4'b1010;//10
         FUNC_SLL : ALU_OP = 4'b1111;//14
         FUNC_SRL : ALU_OP = 4'b1110;//15
```

Endcase
若 ALUOp_i = 101(ADDI)，ALU_OP = 0010
若 ALUOp_i = 000(BEQ) ，ALU_OP = 0110
若 ALUOp_i = 010(ORI) ，ALU_OP = 0001

若是 R-type 指令 sll、srl、sllv、srlv，FURslt_o = 01
其餘 R-tyoe 指令、ADDI 指令與 ORI 指令 FURslt_o = 00
若是 LUI 指令 FURslt_o = 10

Sign_Extend SE(data_i, data_o):
data_o 前 16bit=data_i
data_o 後 16bit=data_i[15]

Zero_Filled ZF(data_i, data_o)
Data_o 後 16bit = data_i
Data_o 前 16bit = 0;

ALU ALU:助教給的 APPENDIX

Shifter shifter(result, leftRight, shamt, sftSrc):
由 ALU_Ctrl 的輸出信號線 aluop[0]決定 leftRight。
result = sftSrc <</>> shamt

Finished part:

Simple_Single_CPU

Problems you met and solutions:

一開始不知道 shifter 的 leftRight 要怎麼從 aluop 拉線。

Summary:

畫電路圖比較方便 debug。