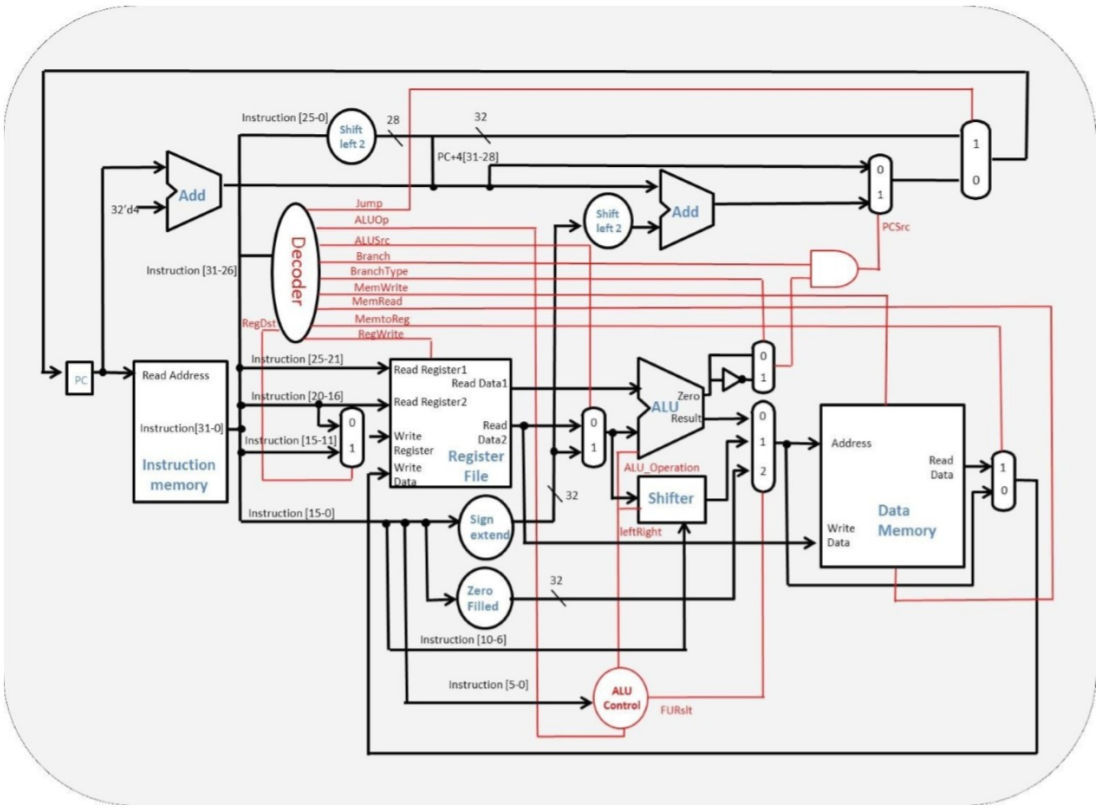
Computer Organization

0617052_0616058

Architecture diagrams:



Hardware module analysis:

ALU_top.v: It's a basic one-bit ALU, including and, or, less, add four operations.

ALU.v: Comprised by 32 ALU_top modules, perform a 32bit ALU. It decodes the ALU control signal and performs the operations which we want.

Instr_Memory.v: It's the instuction memory of CPU.

Program_Counter.v: Pass the 32-b its bus from pc_in_i to pc_out_i during clock edges.

Reg_file.v: All the register datas are inside this module and the 'Reg_Write_i' is used to determine whether the data should be written into the register or not.

TestBench.v: Test the correctness of our design.

Adder.v: A 32-bit adder provides for our PC counter(+4).

Mux2to1.v: A simple 2x1 mux.

Mux3to1.v: A simple 3x1 mux.

ALU_Ctrl.v: Get the ALU_op signals from the decoder and the fuction field (instr [5:0]), then output the ALU_operation for ALU.v and FURslt for 3x1 mux. Compared to the version of Lab2, it has LW, SW, BEQ, BNE... more functions.

Decoder.v: Decode the instruction [31:26] and get the control signals for Reg_Dst, Reg_Write, ALU_op, ALU_Src, Branch_Type, Branch, Jump, Mem_Read, Mem_Write, Mem_to_Reg.

Shifter.v: Shift sftSrc left or right logically.

Sign_Extended.v: Repeat the MSB 16 times in the front of 'data_i '.

Simple_Single_CPU.v: The top module connects all the modules inside the CPU.

Zero_Filled.v: Fill 16 zeros at the back of 'data_i'.

Data_Memory.v: It's the data memory of CPU.

Finished part:

All the instructions in Lab3 pdf.

Problems you met and solutions:

Problem: Advance jal: Set a wrong value of “regwrite”.
Solution: Correct it.

Problem: Advance bgez: Cannot use “Mux3to1” to choose four types.
Solution: Add one more Mux.

Summary:

Understand how to get the control signals of Branch_Type, Branch, Jump, Mem_Read, Mem_Write, Mem_to_Reg by decoding.
Understand how to implement more instructions.