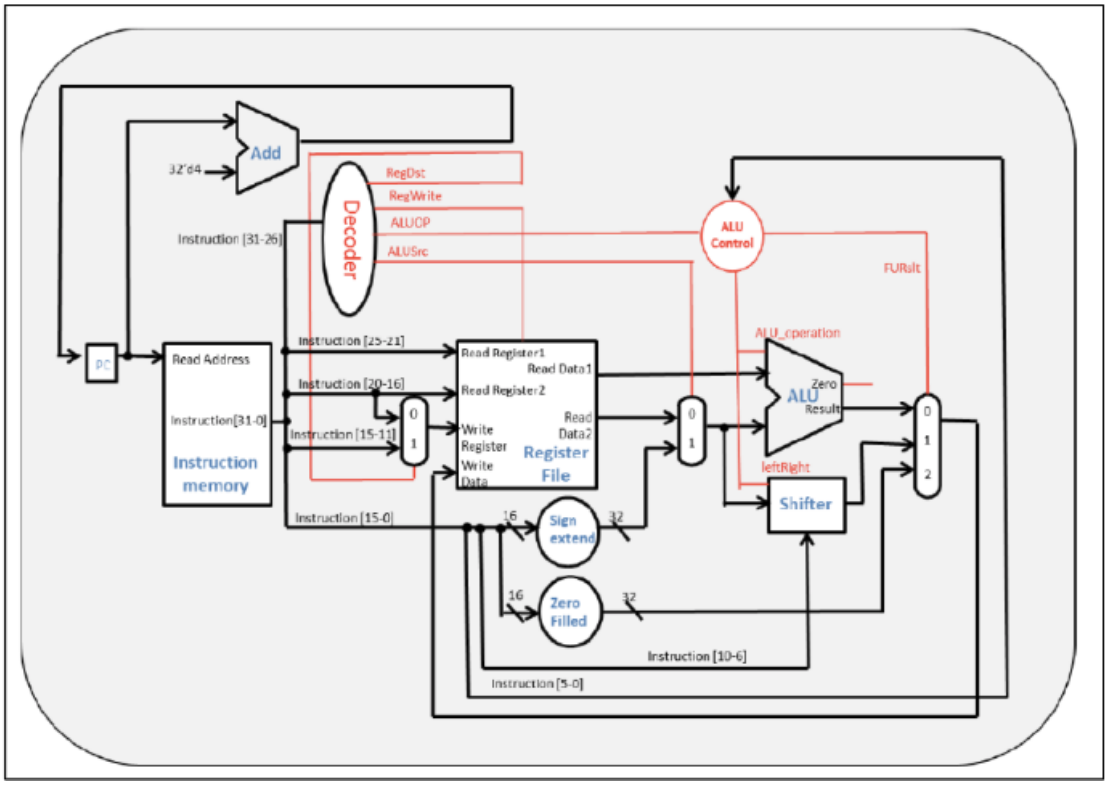
Computer Organization

Architecture diagrams:



Hardware module analysis:

We use the architecture design provided in the PDF to simply finish this lab.

Program Counter (Provided by TA)

- PC is just a 32-bit register where the input is passed to the output triggered by rising edge clock. It is also reset to 0 by a rst signal.

Instruction memory (Provided by TA)

- Instruction memory is a 32x32-bit memory (32 words). Instructions are read and outputted by using PC address from Program Counter.

Adder

- Behavior of Adder is simply input1+input2 = output.

Decoder

- Decoder decode each instruction into various control signals and ALUop based on the instruction opcode and the ISA.

Sign extender

- Sign extender do sign extension to extend a 16-bit data to 32-bit. It is done by taking the sign bit (16^{th} bit) and set it to the 17^{th}-32^{th} bit.

Zero-filled

- This is necessary for LUI instruction. It takes the 16-bit immediate and set it to the upper 16-bit and fill the lower 16-bit with 0.

Mux -2/3 to 1

- Output is chosen from inputs, which the choice is determined by the select wire (1 bit for 2-1, 2bit for 3-1).

Register File (Provided by TA)

- Outputs (RS and RT) is obtained by using the addresses input into the file.
- If RegWrite is set, the input data is assigned to the register specify by RDaddr.

ALU

- ALU is done in last lab, so we do not think analysis is necessary. We also use the MIPS ALU provided by the textbook.

ALU Control

- Taking ALUop from decoder and the funct field of instructions, ALU Control unit determines which operation does ALU need to perform.
- It also determines the final results of the CPU by controlling the control signal of the result Mux. (Choose from ALU, shifter or zero-filled)
- We added 1 more signal/output for the bonus case.

Shifter

- Shift the data left or right depends on the leftright control signal, which the amount it shifts depends on shamt.

Simple Single CPU

- In this module, we linked all components together to form a complete CPU.

Finished part:

All required parts including both basic and bonus case are done.

Problems you met and solutions:

Bonus case: No path for the register to pass into the shamt input for shifter, which is necessary for SRLV, SLLV.

Solution: Add a wire to connect it to the shamt input, and add a 2-1 Mux to select in between shamt from instructions or shamt from register variable.

Summary:

This lab is a good start to learn the details of the CPU design, we implemented an ALU on the previous lab, and during this lab we use other components together with ALU to build a basic CPU, which let us understand more in details on the underlying principle of CPU.