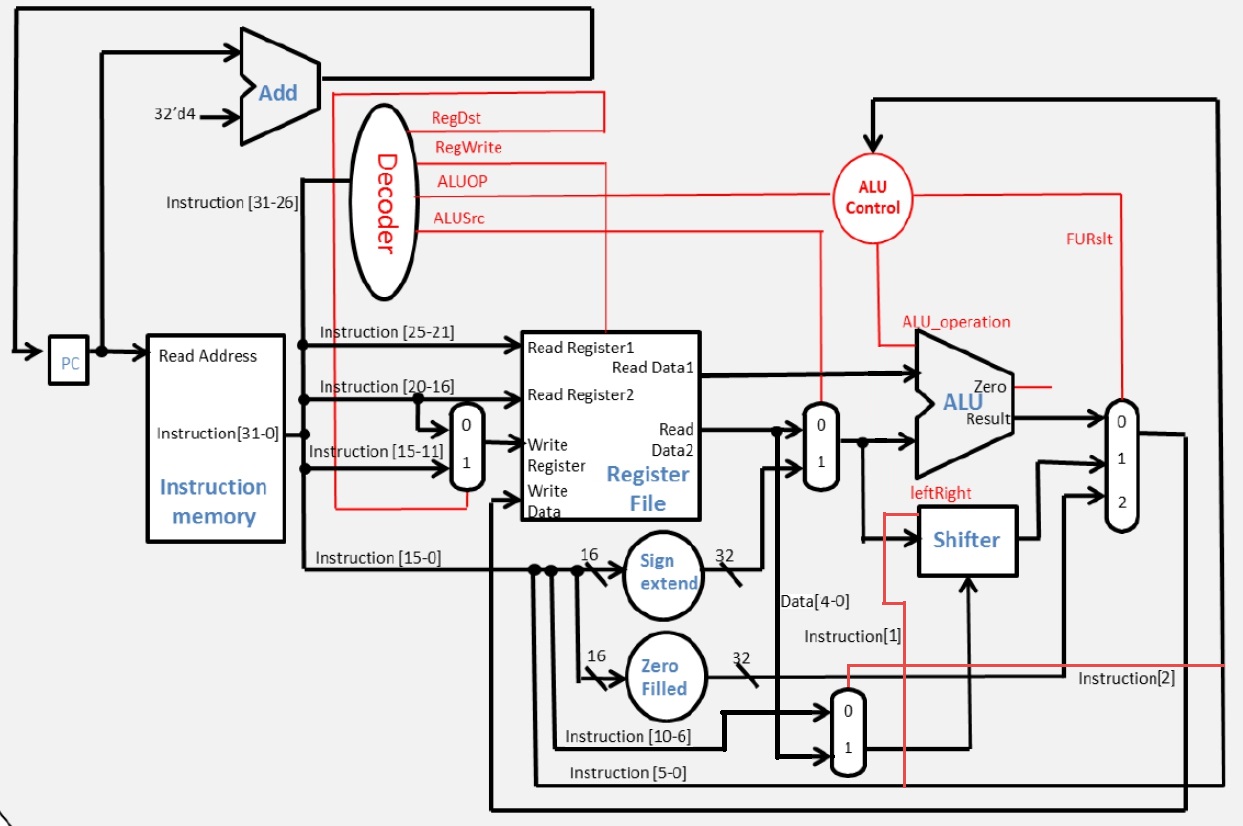
**Computer Organization**

**Student ID: 0416106 Name: 彭敬樺**

**Architecture diagrams:**

****

**Hardware module analysis:**- Adder: This module is used to calculate the sum of 2 31-bit value.  
- ALU: Arithmetic logical unit in a CPU. Used to calculate the result needed for the instruction, such as: add, sub, and, or, nor, slt, addi, and etc.  
- ALU\_1bit: Parts of the module included inside the module “ALU”, used to calculate only 1 bit value.  
- ALU\_bit31: The last bit module in the module “ALU”, usage is quite the same with the normal 1-bit ALU, however, with the additional function to determine zero and overflow.  
- ALU\_Ctrl: Used to give instruction to the ALU, so that it knows what to be calculated from the given machine code. ALU\_Ctrl provides ALU with 4-bit value to maintain its correctness in calculation result.  
- Decoder: Used to control the multiplexer and ALU\_Ctrl to suit the condition needed by the given machine code.  
- Full\_adder: Similar to adder, however, have the capability to receive and return carry.  
- Instr\_Memory: Module used to catch the instruction given its PC.  
- Mux2to1: A multiplexer to choose 1 value from 2 existing choice.  
- Mux3to1: A multiplexer to choose 1 value from 3 existing choice.  
- Program\_Counter: The parts of the module that will maintain the value of the PC so that instruction can progress in the wanted order.  
- Reg\_File: Parts of the Simple\_Single\_CPU, used to get the data in the given address of its storage. It also have the function to write a data into a given address.  
- Shifter: Shift the value given as input left or right as many as the given amount of shamt.  
- Sign\_Extend: Extend the sign bit of the 16-bit value into 32-bit.  
- Simple\_Single\_CPU: The modules that wrap all finished module to gain the wanted result.  
- Zero\_Filled: Fill the lower 16-bit value with 0(zero) and fill the upper 16-bit with input value.

**Finished part:**There are 17 modules used to make this project:  
- Adder.v  
- ALU.v  
- ALU\_1bit.v  
- ALU\_bit31.v  
- ALU\_Ctrl.v  
- Decoder.v  
- Full\_adder.v  
- Instr\_Memory.v  
- Mux2to1.v  
- Mux3to1.v  
- Program Counter.v  
- Reg\_File.v  
- Shifter.v  
- Sign\_Extend.v  
- Simple\_Single\_CPU.v  
- TestBench.v  
- Zero\_Filled.v

**Problems you met and solutions:  
-** Problem: Mistaken the zero\_filled value function as filling the upper 16-bit by 0(zero)  
Solution: Correct it into its supposed usage to fill the lower 16-bit value with 0(zero) and fill the upper 16-bit with input value.  
- Problem: Bit mismatch when connecting one module with another  
Solution: By analyzing the required bit length of the receiving and output module.

**Summary:**

In this project, I have no major problem in completing it. Connecting wire in the module require an elaborate effort to ensure nothing is wrong in the connection. The multiplexer used between some modules has good effect in maintaining and restricting strange value from another unneeded module result from going to next parts. This is especially useful for large module.

In conclusion, this project has let me learn more about the process and logic behind a simple CPU that we have taken as granted nowadays. By understanding the basics of calculation of CPU, then we will be able to understand its limitation and weakness.