B09902123
張偉彥

CA 2022 Fall hw3

Modules Explanation

A. **Control**

It read the opcode to identify these 8 types of instruction into R-type and non R-type. Depending on this identification, we can assign different value of $ALUOp\_o$ and $ALUSrc\_o$ respectively. Moreover, since all types of instruction modify the content of register, $RegWrite\_o$ will always be set to 1.

B. **ALU control**

It decides the computation behavior of ALU based on the value of $funct3$,$funct7$ and $ALUOp\_i$, which comes from Control module. As one of them mentioned above changes, it will reassign the value of $ALUCtrl\_o$, which has 8 types corresponding to 8 different instructions.

C. **Sign extend**

It reads the 12 bits long data and do sign extension based on the the 11th digit of it, which makes it a 32 bits long data.

D. **ALU**

It will read two 32 bits long data and do the required operation according to the value of $ALUCtrl\_i$, which we assign in ALU control module. Moreover, since each time 2 input data or $ALUCtrl\_i$ change it has to take action, I put it in an always block.

E. **MUX32**

It helps us to identify which data should be chosen to go to ALU, and it selects it based on $select\_i$, which is actually $ALUSrc\_o$ from Control module. As I mentioned above, it only divides 8 types of instructions into the group of R-type and the group of remaining. The former requires the data from register2, while the latter requires the data from Sign extend module.

F. **Adder**

Adder module simply add current PC with 4. Hence, it requires current PC value in 32 bits long and reads 4 as a 32 bits long data, too. After that, it just sum them up.

Development Environment

I use Ubuntu 20.04 and iverilog as compiler. Besides, I use VS code to edit it(I think it is useless information.)