Computer Architecture Homework 4 Report

B08902083 謝鈺嘉

I. Modules Explanation

I.1 Adder.v

Adder.v takes two 32-bit inputs, data1_in and data2_in. It has one 32-bit output, data_o. Adder takes the current PC as data1_i and 4 as data2_i to get the next instruction, PC + 4. The next instruction is stored in data_o which is used in PC.v as pc_i.

I.2 ALU.v

ALU.v takes two 32-bit inputs, data1_i and data2_i, and one 3-bit input, ALUCtrl_i. It has one 32-bit output (a register of the same size is also declared), data_o, and one 1-bit output, Zero_o. By using the case statements, ALU.v perform different operation corresponded to ALUCtrl_i as shown in below:

ALUCtrl_i	Instruction	Operation
000	and	data_o <= $signed(data1_i) & $signed(data2_i);
001	xor	data_o <= $signed(data1_i) ^ $signed(data2_i);
010	sll	data_o <= $signed(data1_i) << data2_i;
011	add	data_o <= $signed(data1_i) + $signed(data2_i);
100	sub	data_o <= $signed(data1_i) - $signed(data2_i);
101	mul	data_o <= $signed(data1_i) * $signed(data2_i);
110	addi	data_o <= $signed(data1_i) + $signed(data2_i);
111	srai	data_o <= $signed(data1_i) >>> data2_i[4:0];

The result of the operation is assigned to data_o. In addition to being the result of an operation, data_o is also the write data, RDdata_i in Registers.v.

I.3 ALU_Control.v

ALU_Control.v takes one 10-bit input, funct_i, and one 2-bit input, ALUOp_i. It has one 3-bit output (a register of the same size is also declared), ALUCtrl_o. funct_i is a concatenation of instruction [31:25] and instruction [14:12]. ALUCtrl_o is used in ALU.v to choose which operation is performed. By using case statements, ALU_Control.v assign different values to ALUCtrl_o based on the value of funct_i and ALUOp_i as below:

ALUOp_i	funct_i		Instruction	ALUCtrl_o
	funct_i [9:3]	funct_i [2:0]		
10	0000000	111	and	000
	0000000	100	xor	001
	0000000	001	sll	010
	0000000	000	add	011

	0100000	000	sub	100
	0000001	000	mul	101
00	X	000	addi	110
	X	101	srai	111

I.4 Control.v

Control.v takes two inputs: Op_i which is 7 bits. Op-i corresponds to the opcode of the instruction. Control.v has one 2-bit output (a register of the same size is also declared), ALUOp_o, and two 1-bit output (two registers of the same size is also declared), ALUSrc_o and RegWrite_o. In Control.v, Op_i is used to determine the value of the three outputs based on the table below:

Op_i	ALUOp_o	ALUSrc_o	RegWrite_o
0110011	10	0	1
0010011	00	1	1

ALUOp_o is used to determine which operation is performed by the instruction in ALU_Control.v. Meanwhile, ALUSrc_o is used to determine which data, data1_i or data2_i, is chosen in MUX32.v. On the other hand, RegWrite_o is used to decide whether RDdata_i is written to register of address RDaddr_i.

1.5 CPU.v

CPU.v takes three 1-bit inputs: clk_i, clock of the single cycle datapath, rst_i, used in PC.v as parameter to reset PC to zero, and start_i, used in PC.v as parameter if next PC is pc_i. For those modules in CPU.v, the input ports and output ports are connected as shown in the datapath from the Homework question by a wire declared with corresponded size between two end.

I.6 MUX32.v

MUX32.v takes two 32-bit inputs, data1_i and data2_i, and one 1-bit input, select_i. It has one 32-bit output, data_o. MUX32 chooses data which data to output based on select_i. If the value of select_i is zero, then data1_i would be returned. Likewise, if select_i is one, data2_i would be assigned as output. MUX32 is used to select between data from register and result of sign extension as the second input, data2_i, for the ALU.

I.7 Sign_Extend.v

Sign_Extend.v takes one 12-bit inputs, data_i. It has one 32-bit output, data_o. The data_i is signed extended by using the concatenation operator of Verilog. It takes the sign bit, data_i[11], and copies it 20 times in front of the input data, data_i[11:0] to fill in 32 bits. The extended value is stored in data_o which is used as the second input, data2_i, in MUX32.

II. Development Environment

II.1 Operating System： Linux (CSIE Workstation)

II.2 Compiler : iverilog