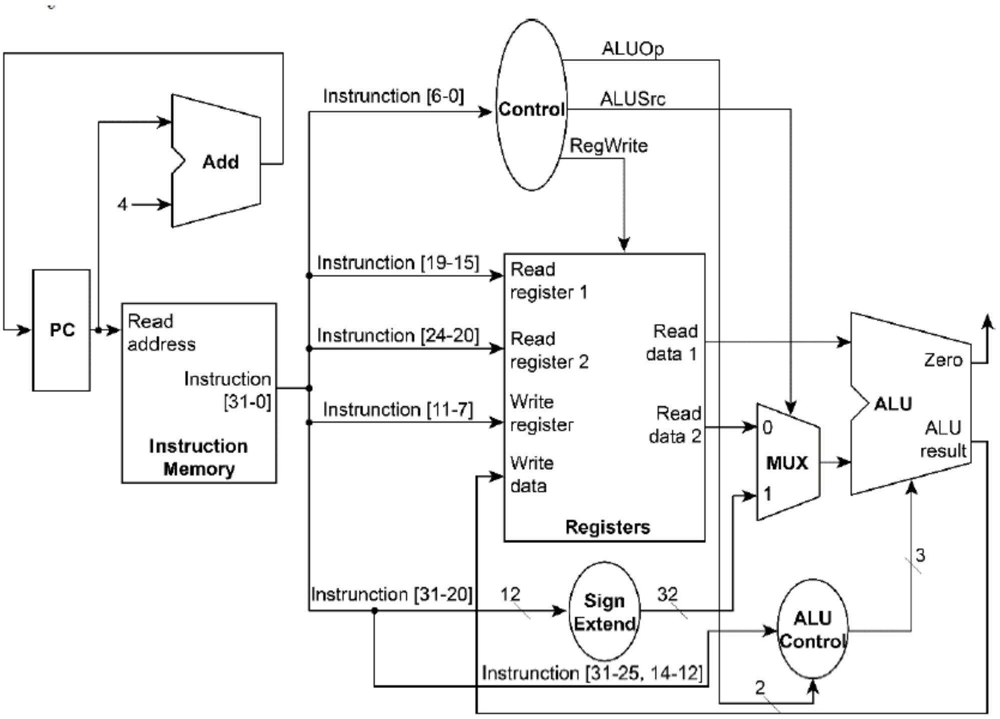
Computer Architecture Hw4 Report B07902091 周俊廷

2.1 Module Explanation

<table>
<tr><td>Adder.v</td><td>Adder.v module have two inputs, data1_in and data2_in, both are 32 bits, one output, data_o which is also 32 bits.
In Adder.v, data1_in and data2_in will be added and the result is assigned to data_o.
The data_o is used in PC.v as pc_i which contain the value of current PC + 4</td></tr>
<tr><td>ALU.v</td><td>ALU.v module have three inputs, data1_i and data2_i are 32 bits while ALUCtrl_i is 3 bits, two outputs, data_o which is 32 bits and Zero_o which is 1 bit.
A register of size 32 bits is declared for data_o.
By using the case statements, ALU.v perform different operation corresponded to ALUCtrl_i as shown in the table below:
<table>
<tr><th>ALUCtrl_i</th><th>Instr</th><th>Operation</th></tr>
<tr><td>000</td><td>and</td><td>data_o <= $signed(data1_i) & $signed(data2_i);</td></tr>
<tr><td>001</td><td>xor</td><td>data_o <= $signed(data1_i) ^ $signed(data2_i);</td></tr>
<tr><td>010</td><td>sll</td><td>data_o <= $signed(data1_i) << data2_i;</td></tr>
<tr><td>011</td><td>add</td><td>data_o <= $signed(data1_i) + $signed(data2_i);</td></tr>
<tr><td>100</td><td>sub</td><td>data_o <= $signed(data1_i) - $signed(data2_i);</td></tr>
<tr><td>101</td><td>mul</td><td>data_o <= $signed(data1_i) * $signed(data2_i);</td></tr>
<tr><td>110</td><td>addi</td><td>data_o <= $signed(data1_i) + $signed(data2_i);</td></tr>
<tr><td>111</td><td>srai</td><td>data_o <= $signed(data1_i) >>> data2_i[4:0];</td></tr>
</table>
The result of the operation is assigned to data_o.
data_o is the result of an instruction and also the write data, RDdata_i in Registers.v.</td></tr>
<tr><td>Sign_Extend.v</td><td>Sign_Extend.v module have one input, data_i which is 12 bits and one output,data_o which is 32 bits.
The 12 bits data_i is signed extended by using the concatenation operator of Verilog.
Data_o = $signed({{20{data_i[11]}},data_i[11:0]})
The data_o is used in MUX32.v as the second input data, data2_i.</td></tr>
</table>

<table>
<tr>
<td>MUX32.v</td>
<td>
<b>MUX32.v</b> module have 3 inputs, <b>data1_i</b> and <b>data2_i</b> which both 32 bits and <b>select_i</b> which is 1 bits, one output, <b>data_o</b> which is 32 bits.
In <b>MUX32.v</b>, the <b>select_i</b> is used to decide which input data is assign to <b>data_o</b>.
<table>
<tr><th>select_i</th><th>data_o</th></tr>
<tr><td>0</td><td>data1_i</td></tr>
<tr><td>1</td><td>data2_i</td></tr>
</table>
The <b>data_o</b> is used in <b>ALU.v</b> as the second input data, <b>data2_i</b> for operation.
</td>
</tr>
<tr>
<td>CPU.v</td>
<td>
<b>CPU.v</b> modeule have three inputs, <b>clk_i</b>, <b>rst_i</b> and <b>start_i</b> which all in 1 bit.
<b>clk_i</b> is the clock of the single cycle datapath
<b>rst_i</b> is used in <b>PC.v</b> to decide whether reset the PC to 0.
<b>start_i</b> is used in <b>PC.v</b> to decide whether next PC is <b>pc_i</b>.
A wire, <b>Four</b> of size 32 bits is declared and assign 32’d4 to be used as the input for <b>data2_i</b> in Add_PC().
A wire, <b>Zero</b> of size 1 bit is declared and used to receive the value of <b>Zero_o</b> in ALU().
For those modules in <b>CPU.v</b>, the input ports and output ports are connected as shown in the datapath diagram below by a wire declared with corresponded size between two end.


</td>
</tr>
</table>

<table>
<tr>
<td>Control.v</td>
<td>
Control.v module have one input, Op_i which is 7 bits and three outputs, ALUOp_o which is 2 bits, ALUSrc_o and RegWrite_o which both 1 bit.
The Op_i is the opcode of an instruction.
Three registers are declared for three outputs with corresponding size.
In Control.v, Op_i is used to determined the value of three outputs based on the table below:
<table>
<tr><th>Op_i</th><th>ALUOp_o</th><th>ALUSrc_o</th><th>RegWrite_o</th></tr>
<tr><td>0110011</td><td>10</td><td>0</td><td rowspan="2">1</td></tr>
<tr><td>0010011</td><td>00</td><td>1</td></tr>
</table>
ALUOp_o is used in ALU_Control.v to determine which operation to be performed by the instruction
ALUSrc_o is used in MUX32.v to determine either data1_i or data2_i is choose.
RegWrite_o is used in Registers.v to decide whether Write Data, RDdata_i is written to register of address RDaddr_i.
</td>
</tr>
<tr>
<td>ALU_Control.v</td>
<td>
ALU_Control.v module have two inputs, funct_i which is 10 bits and ALUOp_i which is 2 bits, one output, ALUCtrl_o which is 3 bits.
funct_i is a concatenation of instruction [31:25] and instruction [14:12].
A register of size 3 bits is declared for ALUCtrl_o.
By using case statements and if-else statements, ALU_Control.v assign different value to ALUCtrl_o based on the value of funct_i and ALUOp_i.
<table>
<tr><th rowspan="2">ALUOp_i</th><th colspan="2">funct_i</th><th rowspan="2">Instruction</th><th rowspan="2">ALUCtrl_o</th></tr>
<tr><th>funct_i [9:3]</th><th>funct_i [2:0]</th></tr>
<tr><td rowspan="6">10</td><td>0000000</td><td>111</td><td>and</td><td>000</td></tr>
<tr><td>0000000</td><td>100</td><td>xor</td><td>001</td></tr>
<tr><td>0000000</td><td>001</td><td>sll</td><td>010</td></tr>
<tr><td>0000000</td><td>000</td><td>add</td><td>011</td></tr>
<tr><td>0100000</td><td>000</td><td>sub</td><td>100</td></tr>
<tr><td>0000001</td><td>000</td><td>mul</td><td>101</td></tr>
<tr><td rowspan="2">00</td><td>X</td><td>000</td><td>addi</td><td>110</td></tr>
<tr><td>X</td><td>101</td><td>srai</td><td>111</td></tr>
</table>
ALUCtrl_o is used in ALU.v to determine which operation to be performed.
</td>
</tr>
</table>

2.2 Module Explanation

The OS used : CSIE Workstation

The Compiler used : iverilog