Computer Architecture
HW4

B03901116 Danny Wang

1 Coding Environment

I compile my codes using *iverilog* on x86 computer with Arch Linux OS.

2 Module Implementation I

Four modules below are more easier and without thinking.

1. CPU.v is most important role of this homework!
   Declare lots of wires with correct width to accomplish this module.

2. MUX5.v, MUX32.v

```
assign data_o = select_i ? data2_i : data1_i;
```

3. Adder.v

```
assign data_o = data1_in + data2_in;
```

4. Sign_Extend.v
   Sign extending is just extending MSB.

```
assign data_o = {{16{data_i[15]}}, data_i};
```

3 Module Implementation II

1. Control.v

```
// assign TYPE values for instructions
assign r_type = (Op_i == 6'b0) ? 1 : 0;
assign addi   = (Op_i == 6'b00_1000) ? 1 : 0;
// assign SIGNAL values for use
assign RegDst_o   = (Op_i == 6'b0) ? 1 : 0;
assign ALUOp_o    = r_type ? RTYPE :
                    addi   ? ADD : 2'b00;
assign ALUSrc_o   = addi;
assign RegWrite_o = r_type | addi;
```

2. ALU_Control.v

```
assign ALUCtrl_o =
  ALUOp_i == 2'b11 ?
    ( funct_i[3:0] == 4'b0000 ? ADD :
//... funct_i[3:0] == <four bits> ? <op-type> :
      funct_i[3:0] == 4'b1000 ? MUL : 3'b0 ) :
    ALUOp_i == 2'b00 ? ADD :
    ALUOp_i == 2'b01 ? SUB :
    ALUOp_i == 2'b10 ? OR  : 3'b0;
```

3. ALU.v

```
assign zero_o = (data1_i == data2_i) ? 1 : 0;
always@ (*) begin
  case(ALUCtrl_i)
    AND: data_o = data1_i & data2_i;
    OR : data_o = data1_i | data2_i;
    ADD: data_o = data1_i + data2_i;
    SUB: data_o = data1_i - data2_i;
    MUL: data_o = data1_i * data2_i;
    default data_o = 32'b0;
  endcase
end
```