module alu( A, B, ALU\_Sel, ALU\_Result);

input [7:0] A,B; // ALU 8-bit Inputs

input [3:0] ALU\_Sel;// ALU Selection

output [8:0] ALU\_Result;// ALU 8-bit Result

reg [8:0] ALU\_Result;

always @(\*) begin

case(ALU\_Sel)

4'b0000:ALU\_Result = A + B ; // Addition

4'b0001:ALU\_Result = A - B ; // Subtraction

//4'b0010:ALU\_Result = A \* B; // Multiplication

4'b0011: ALU\_Result = A/B; // Division

4'b0100: ALU\_Result = A<<1;// Logical shift left

4'b0101: ALU\_Result = A>>1;// Logical shift right

4'b0110: ALU\_Result = {A[6:0],A[7]};// Rotate left

4'b0111: ALU\_Result = {A[0],A[7:1]}; // Rotate right

4'b1000: ALU\_Result = A & B; // Logical and

4'b1001: ALU\_Result = A | B;// Logical or

4'b1010: ALU\_Result = A ^ B;// Logical xor

4'b1011: ALU\_Result = ~(A | B);// Logical nor

4'b1100: ALU\_Result = ~(A & B); // Logical nand

4'b1101: ALU\_Result = ~(A ^ B); // Logical xnor

4'b1110: ALU\_Result = (A>B)?8'd1:8'd0 ; // Greater comparison

4'b1111: ALU\_Result = (A==B)?8'd1:8'd0 ;// Equal comparison

default: ALU\_Result = 0;

endcase

end

endmodule