```
module ALUtb#(parameter W=4);

reg clktb;
reg [3:0] Atb;
reg [3:0] Btb;
reg [2:0] ALUcntrltb;
reg [3:0] Oexp;
reg COexp;
reg OVFexp;
reg Nexp;
reg Zexp;
reg [31:0] vectornum,errors;
reg [18:0] testvectors[9:0];
wire [3:0] Otb;
wire COtb;
wire OVFtb;
wire Ntb;
wire Ztb;

ALU dut(.A(Atb),.B(Btb),.ALUcntrl(ALUcntrltb),
         .O(Otb),.CO(COtb),.OVF(OVFtb),.N(Ntb),.Z(Ztb));
// To generate clock cycles
always
begin
   clktb=~clktb;
   #5;
end

// To read vectors
initial
begin
   clktb=1;
   $readmemb("ALUtb.tv",testvectors);
   vectornum = 0;
   errors = 0;
end
// Test Vectors are read and operated at pos edges
   always @(posedge clktb)
   begin
      #1;
      {Atb,Btb,ALUcntrltb,Oexp,COexp,OVFexp,Nexp,Zexp}=testvectors[vectornum];
   end
// Test Vector Output and Expect are compared at neg edges
   always @(negedge clktb)
   begin
      if(Otb != Oexp)
      begin
         $display("Error: input = %b %b", Atb, Btb);
         $display("O outputs = %b (%b expected)",Otb,Oexp);
         errors= errors +1;
      end
      if(COtb != COexp)
      begin
         $display("Error: input = %b %b", Atb, Btb);
         $display("CO outputs = %b (%b expected)",COtb,COexp);
         errors= errors +1;
      end
      if(OVFtb != OVFexp)
      begin
         $display("Error: input = %b %b", Atb, Btb);
         $display("OVF outputs = %b (%b expected)",OVFtb,OVFexp);
         errors= errors +1;
      end
      if(Ntb != Nexp)
      begin
         $display("Error: input = %b %b", Atb, Btb);
```

```
67            $display("N outputs = %b (%b expected)",Ntb,Nexp);
68            errors= errors +1;
69         end
70         if(Ztb != Zexp)
71         begin
72            $display("Error: input = %b %b", Atb, Btb);
73            $display("Z outputs = %b (%b expected)",Ztb,Zexp);
74            errors= errors +1;
75         end
76   // Increment array index
77         vectornum = vectornum +1;
78         if(testvectors[vectornum] == 4'bx)
79         begin
80            $display("%d tests completed with %d errors",vectornum, errors);
81            $stop; // End simulation
82         end
83      end
84   endmodule
85
86
87
88
89
90
91
92
93
94
95
96
97
```