CODE

```
module led5(
    input [5:0] letter,  // 6-bit input for 26 letters (A-Z)
    output reg [15:0] segments // 16-segment display output (a-p)
);
// Segment encoding for a 16-segment display
// segments[15:0] -> {N, M, L, K, J, H, G2, G1, F, E, D2, D1, C, B, A2, A1}
// Each bit corresponds to a segment on the 16-segment display
always @(*) begin
    case (letter)
            // Uppercase Alphabets
                6'd0:  segments = 16'b0000_0000_0000_0000; // null
        6'd1:  segments = 16'b0000_0011_1100_1111; // A
        6'd2:  segments = 16'b0100_1010_0011_1111; // B
        6'd3:  segments = 16'b0000_0000_1111_0011; // C
        6'd4:  segments = 16'b0100_1000_0011_1111; // D
        6'd5:  segments = 16'b0000_0011_1111_0011; // E
        6'd6:  segments = 16'b0000_0001_1110_0011; // F
        6'd7:  segments = 16'b0000_0010_1111_1011; // G
        6'd8:  segments = 16'b0000_0011_1100_1100; // H
        6'd9:  segments = 16'b0100_1000_0011_0011; // I
        6'd10: segments = 16'b0100_1000_0110_0011; // J
        6'd11: segments = 16'b0011_0001_1100_0000; // K
        6'd12: segments = 16'b0000_0000_1111_0000; // L
        6'd13: segments = 16'b0101_0100_1100_1100; // M
        6'd14: segments = 16'b0010_0100_1100_1100; // N
        6'd15: segments = 16'b0000_0000_1111_1111; // O
        6'd16: segments = 16'b0000_0011_1100_0111; // P
        6'd17: segments = 16'b0010_0000_1111_1111; // Q
        6'd18: segments = 16'b0010_0011_1100_0111; // R
        6'd19: segments = 16'b0000_0011_1011_1011; // S
```

```
6'd20: segments = 16'b0100_1000_0000_0011; // T
6'd21: segments = 16'b0000_0000_1111_1100; // U
6'd22: segments = 16'b1001_0000_1100_0000; // V
6'd23: segments = 16'b1010_1000_1100_1100; // W
6'd24: segments = 16'b1011_0100_0000_0000; // X
6'd25: segments = 16'b0100_0011_1000_0100; // Y
6'd26: segments = 16'b1001_0000_0011_0011; // Z
// lowercase Alphabet
6'd27: segments = 16'b0000_0011_0111_1111; // a
6'd28: segments = 16'b0000_0011_1111_1000; // b
6'd29: segments = 16'b0000_0011_0111_0000; // c
6'd30: segments = 16'b0000_0011_0111_1100; // d
6'd31: segments = 16'b0000_0011_1111_0111; // e
6'd32: segments = 16'b0100_1011_0000_0010; // f
6'd33: segments = 16'b0000_0011_1011_1111; // g
6'd34: segments = 16'b0000_0011_1100_1000; // h
6'd35: segments = 16'b0100_0001_0011_0001; // i
6'd36: segments = 16'b0000_0000_0011_1011; // j
6'd37: segments = 16'b0010_0011_1100_0000; // k
6'd38: segments = 16'b0100_1000_0010_0001; // l
6'd39: segments = 16'b0100_0011_0100_1000; // m
6'd40: segments = 16'b0000_0000_1100_1111; // n
6'd41: segments = 16'b0000_0011_0111_1000; // o
6'd42: segments = 16'b0001_0001_1100_0011; // p
6'd43: segments = 16'b0001_0001_1000_1111; // q
6'd44: segments = 16'b0000_0011_0100_0000; // r
6'd45: segments = 16'b0010_0010_0011_0000; // s
6'd46: segments = 16'b0100_1011_0010_0000; // t
6'd47: segments = 16'b0000_0000_0111_1000; // u
6'd48: segments = 16'b1000_0000_0100_0000; // v
6'd49: segments = 16'b1010_0000_0100_1000; // w
```

```
        6'd50: segments = 16'b1011_0100_0000_0000; // x
        6'd51: segments = 16'b0000_1010_0011_1100; // y
        6'd52: segments = 16'b1000_0001_0001_0000; // z
        default: segments = 16'b1111_1111_1111_1111; // Blank or invalid input
    endcase
end
endmodule
```

TEST BENCH CODE

```
module led5_tb_v;
            // Inputs
            reg [5:0] letter;
            // Outputs
            wire [15:0] segments;
            // Instantiate the Unit Under Test (UUT)
            led5 uut (
                        .letter(letter),
                        .segments(segments));
            initial begin
        // Initialize letter input
        letter = 0; // Start with A
        // Loop through all 26 letters (A-Z)
        repeat(52) begin
            #10;                  // Wait 10 time units
            $display("Letter: %c, Segments: %b", letter + 65, segments);
            letter = letter + 1;   // Increment to the next letter
        end
        #10;
        $finish;
    End
endmodule
```