Always nolatches ○

Suppose you're building a circuit to process scancodes from a PS/2 keyboard for a game. Given the last two bytes of scancodes received, you need to indicate whether one of the arrow keys on the keyboard have been pressed. This involves a fairly simple mapping, which can be implemented as a case statement (or if-elseif) with four cases.

Scancode [15:0]	Arrow key
`16'he06b`	left arrow
`16'he072`	down arrow
`16'he074`	right arrow
`16'he075`	up arrow
Anything else	none

Your circuit has one 16-bit input, and four outputs. Build this circuit that recognizes these four scancodes and asserts the correct output.

To avoid creating latches, all outputs must be assigned a value in all possible conditions (See also always_if2 ✓). Simply having a `default` case is not enough. You must assign a value to all four outputs in all four cases and the default case. This can involve a lot of unnecessary typing. One easy way around this is to assign a "default value" to the outputs *before* the case statement:

```
always @(*) begin
    up = 1'b0; down = 1'b0; left = 1'b0; right = 1'b0;
    case (scancode)
        ... // Set to 1 as necessary.
    endcase
end
```

This style of code ensures the outputs are assigned a value (of 0) in all possible cases unless the case statement overrides the assignment. This also means that a `default:` case item becomes unnecessary.

Reminder: The logic synthesizer generates a combinational circuit that *behaves* equivalently to what the code describes. Hardware does not "execute" the lines of code in sequence.