Your questions all about...

What I have to do...?

How can I do...?
What you have to do!
What you have to do! (Part B)

module hazard (H_ID_rs, H_ID_rt, H_EX_rs, H_EX_rt, H_EX_mr, H_MEM_RegWrite, H_WB_RegWrite, H_EX_RegRd, H_MEM_RegRd, H_WB_RegRd, H_MEM_MemtoReg, H_ID_PCsrc, H_ID_PCJump, H_ID_Branch, H_EX_RegWrite, H_Stall, H_Flush, H_FowardAE, H_FowardBE, H_FowardAD, H_FowardBD);

input [4:0] H_ID_rs, H_ID_rt, H_EX_rs, H_EX_rt, H_EX_mr, H_MEM_RegRd, H_WB_RegRd;
input H_MEM_RegWrite, H_WB_RegWrite, H_EX_RegWrite, H_ID_Branch, H_EX_MemtoReg, H_MEM_MemtoReg, H_ID_PCsrc, H_ID_PCJump;
output reg [1:0] H_FowardAE, H_FowardBE;
output reg H_FowardAD, H_FowardBD;
output reg H_Stall, H_Flush;

initial begin
    H_FowardAE = 2’b00;
    H_FowardBE = 2’b00;
    H_FowardAD = 1’b0;
    H_FowardBD = 1’b0;
    H_Stall = 1’b0;
    H_Flush = 1’b0;
end
endmodule

Format of inputs

[H][Stage where input used][Description of input]

- ForwardAD, ForwardBD : Forward from Mem stage
- ForwardAE, ForwardBE : Forward from Mem or WB stage
- e.g. ForwardAE = 2’b00 => not hazard
  ForwardAE = 2’b01 => Forward from WB
  ForwardAE = 2’b10 => Forward from Mem
What you have to do! (Part C)

branch_predictor module(branch_prediction.v)

```verilog
module branch_prediction_v

input [31:0] WB指令;
input [5:0] IF指令code;
input [3:0] WB_bp_offset, IF_pc_offset;
input clk, WB_branch, WB_branch_taken, halt;
output reg [31:0] IF_pc_out;
output reg [3:0] IF_bp_offset;
output reg IF_prdTaken;
reg [1:0] prediction_buffer [0:15];
reg [31:0] target_buffer [0:15];
integer fp, i;

// Do not delete: initialize the predictor and the target buffer initial

begin
for (i = 0; i < 16; i++)
begin
    prediction_buffer [i] = 2'b01;
    target_buffer [i] = 32'b0;
end
end
```

<table>
<thead>
<tr>
<th>prediction_buffer</th>
<th>target_buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>pc to jump(branch inst. 1)</td>
<td>pc to jump(branch inst. 1)</td>
</tr>
<tr>
<td>pc to jump(branch inst. 2)</td>
<td>pc to jump(branch inst. 2)</td>
</tr>
<tr>
<td>pc to jump(branch inst. 3)</td>
<td>pc to jump(branch inst. 3)</td>
</tr>
<tr>
<td>pc to jump(branch inst. 4)</td>
<td>pc to jump(branch inst. 4)</td>
</tr>
<tr>
<td>pc to jump(branch inst. 5)</td>
<td>pc to jump(branch inst. 5)</td>
</tr>
</tbody>
</table>
What you have to do! (Part C) – cont.

---

```verbatim
// Part C: output prediction and target address
always @(IF_pc_offset or IF_opcode)
begin

end
```

```verbatim
// WRITE YOUR CODES BELOW!!!!!!!!!!!!
// Please write your code to make branch predictor work properly

endmodule
```

---

```verbatim
// Part C: update the predictor and target buffer
always @(negedge clk)
begin

end
```

```verbatim
// WRITE YOUR CODES BELOW!!!!!!!!!!!!
// Please write your code to make branch predictor work properly

endmodule
```
What you have to do! (Part C) – cont.

```verilog
module misprediction_control
(input clk, predTaken, calcTaken, tgtCorrect, PCsrc, TgtSrc, isBranch, stat);
output reg PCsrc, TgtSrc;

// Part C: misprediction controller
always @ (predTaken or calcTaken or tgtCorrect)
begin

==============================================

//=================================================================

// WRITE YOUR CODES BELOW!!!!!!!!!!!!
//=================================================================

Please write your code to make misprediction controller work properly.

==============================================

PCsrc = calcTaken;
    TgtSrc = isBranch;

end

endmodule
```