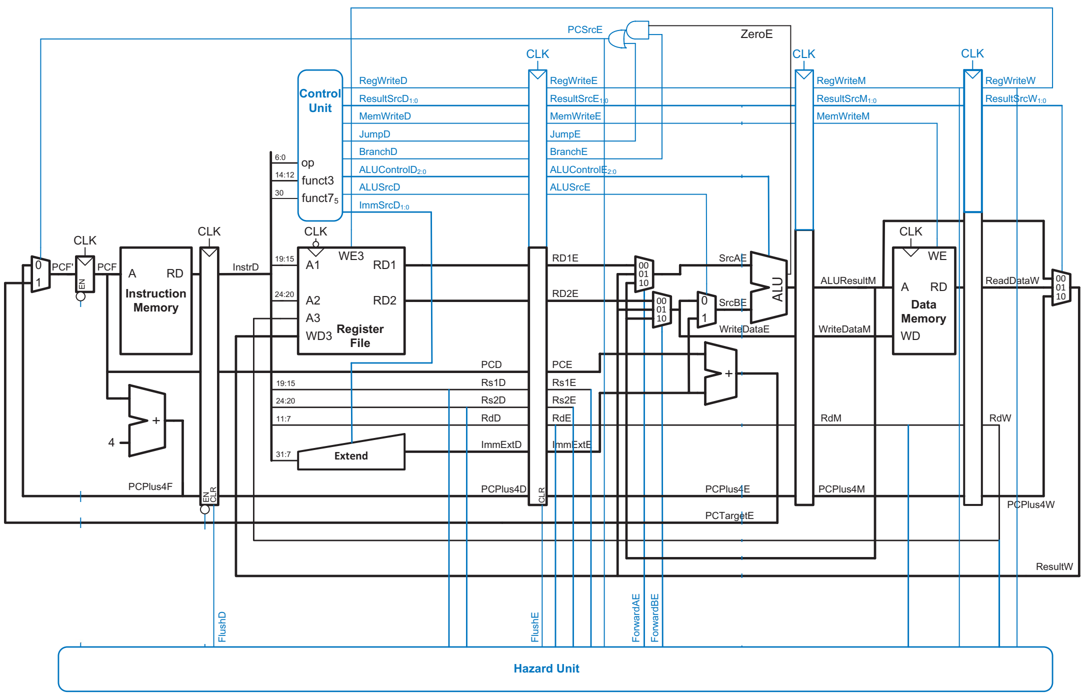

Figure 7.61 Pipelined processor with full hazard handling

Forward to solve data hazards when possible[3]**:**

if $((Rs1E == RdM)\ \&\ RegWriteM)\ \&\ (Rs1E\ != 0)$ then

$ForwardAE = 10$

else if $((Rs1E == RdW)\ \&\ RegWriteW)\ \&\ (Rs1E\ != 0)$ then

$ForwardAE = 01$

else $ForwardAE = 00$

Flush when a branch is taken or a load introduces a bubble:

$FlushD = PCSrcE$

$FlushE = \quad PCSrcE$

[3] Recall that the forwarding logic for *SrcBE* (*ForwardBE*) is identical except that it checks *Rs2E* instead of *Rs1E*.

Exercise 7.45

SystemVerilog

```
module hazard(input  logic [4:0]             Rs1E, Rs2E,      RdM, RdW,
              input  logic       PCSrcE,
              input  logic       RegWriteM, RegWriteW,
              output logic [1:0] ForwardAE, ForwardBE,
              output logic                          FlushD, FlushE);



  // forwarding logic
  always_comb begin
    ForwardAE = 2'b00;
    ForwardBE = 2'b00;
    if (Rs1E != 5'b0)
      if      ((Rs1E == RdM) & RegWriteM) ForwardAE = 2'b10;
      else if ((Rs1E == RdW) & RegWriteW) ForwardAE = 2'b01;

    if (Rs2E != 5'b0)
      if      ((Rs2E == RdM) & RegWriteM) ForwardBE = 2'b10;
      else if ((Rs2E == RdW) & RegWriteW) ForwardBE = 2'b01;
  end

  //            flushes



  assign FlushD   = PCSrcE;
  assign FlushE   =            PCSrcE;
endmodule
```