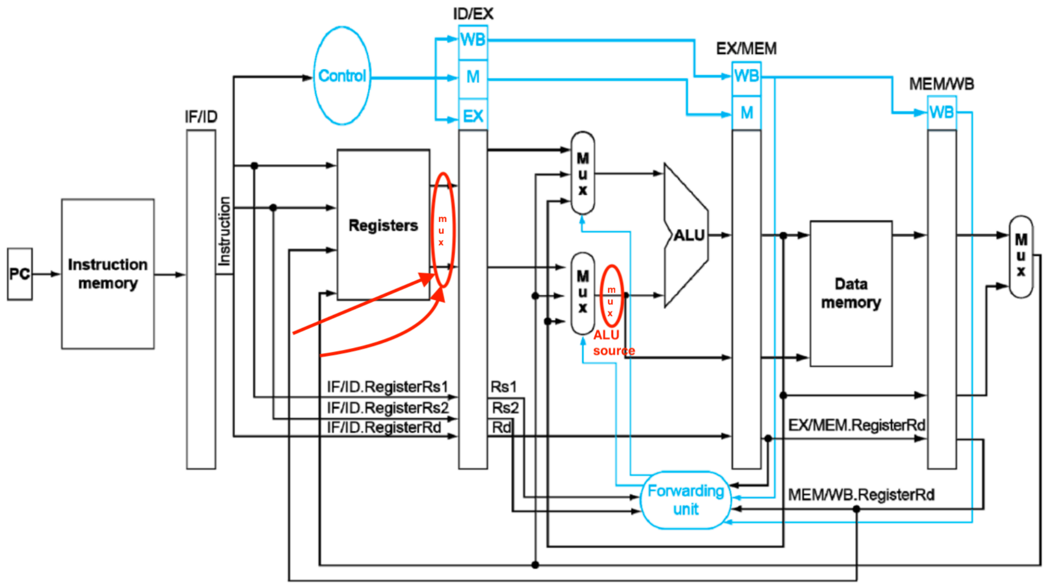
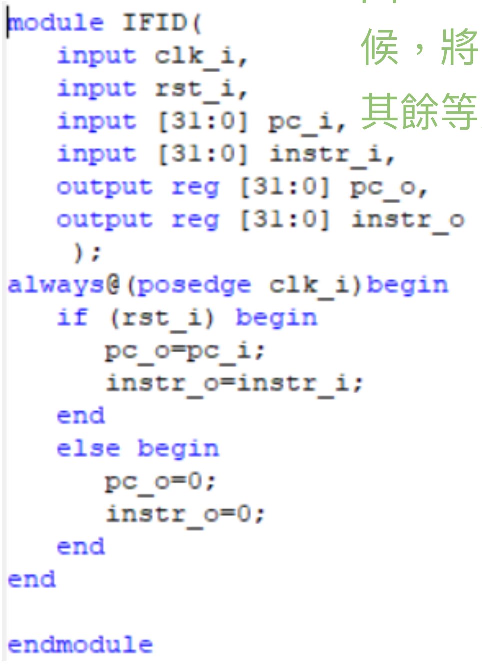
Computer Organization

Architecture diagram:



Detailed description of the implementation:

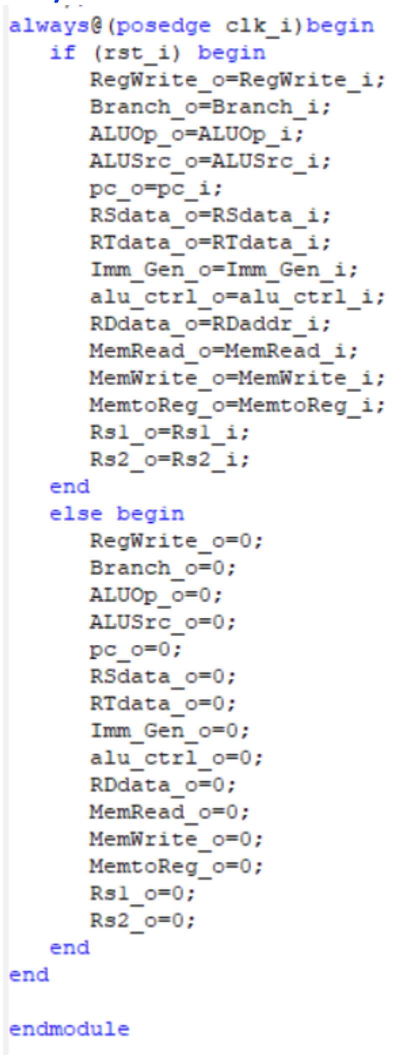
IF/ID:



```
module IFID(
    input clk_i,
    input rst_i,
    input [31:0] pc_i,
    input [31:0] instr_i,
    output reg [31:0] pc_o,
    output reg [31:0] instr_o
     );
always@(posedge clk_i)begin
    if (rst_i) begin
        pc_o=pc_i;
        instr_o=instr_i;
    end
    else begin
        pc_o=0;
        instr_o=0;
    end
end

endmodule
```

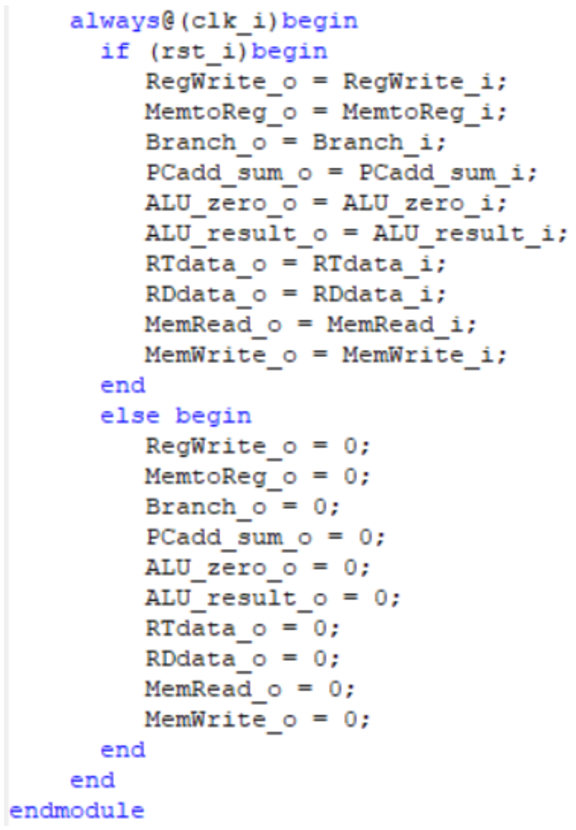
pipeline: 當rst_i等於1的時候，將input傳給output，其餘等於0時歸零

ID/EX:



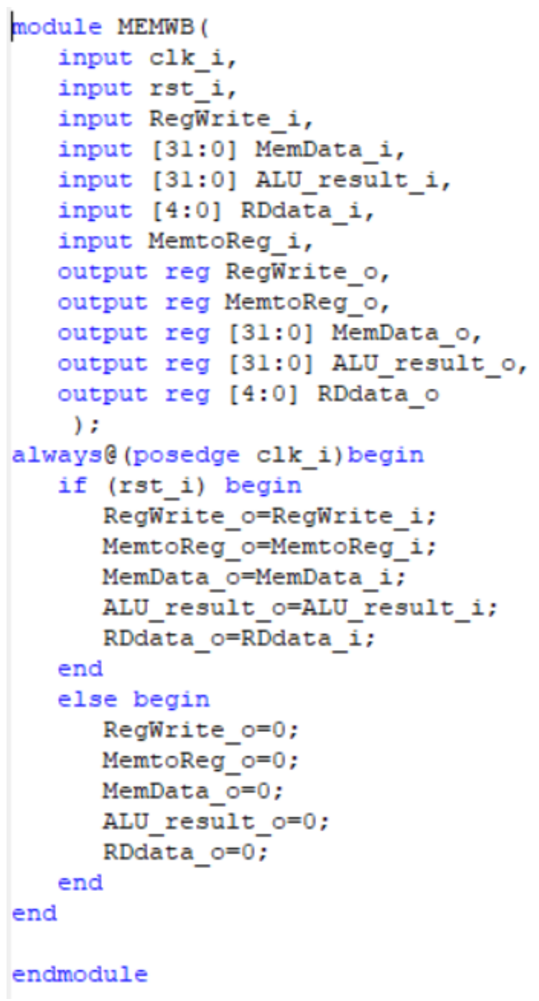
```
always@(posedge clk_i)begin
    if (rst_i) begin
        RegWrite_o=RegWrite_i;
        Branch_o=Branch_i;
        ALUOp_o=ALUOp_i;
        ALUSrc_o=ALUSrc_i;
        pc_o=pc_i;
        RSdata_o=RSdata_i;
        RTdata_o=RTdata_i;
        Imm_Gen_o=Imm_Gen_i;
        alu_ctrl_o=alu_ctrl_i;
        RDdata_o=RDaddr_i;
        MemRead_o=MemRead_i;
        MemWrite_o=MemWrite_i;
        MemtoReg_o=MemtoReg_i;
        Rs1_o=Rs1_i;
        Rs2_o=Rs2_i;
    end
    else begin
        RegWrite_o=0;
        Branch_o=0;
        ALUOp_o=0;
        ALUSrc_o=0;
        pc_o=0;
        RSdata_o=0;
        RTdata_o=0;
        Imm_Gen_o=0;
        alu_ctrl_o=0;
        RDdata_o=0;
        MemRead_o=0;
        MemWrite_o=0;
        MemtoReg_o=0;
        Rs1_o=0;
        Rs2_o=0;
    end
end

endmodule
```

EX/MEM:



```
    always@(clk_i)begin
      if (rst_i)begin
         RegWrite_o = RegWrite_i;
         MemtoReg_o = MemtoReg_i;
         Branch_o = Branch_i;
         PCadd_sum_o = PCadd_sum_i;
         ALU_zero_o = ALU_zero_i;
         ALU_result_o = ALU_result_i;
         RTdata_o = RTdata_i;
         RDdata_o = RDdata_i;
         MemRead_o = MemRead_i;
         MemWrite_o = MemWrite_i;
      end
      else begin
         RegWrite_o = 0;
         MemtoReg_o = 0;
         Branch_o = 0;
         PCadd_sum_o = 0;
         ALU_zero_o = 0;
         ALU_result_o = 0;
         RTdata_o = 0;
         RDdata_o = 0;
         MemRead_o = 0;
         MemWrite_o = 0;
      end
    end
endmodule
```

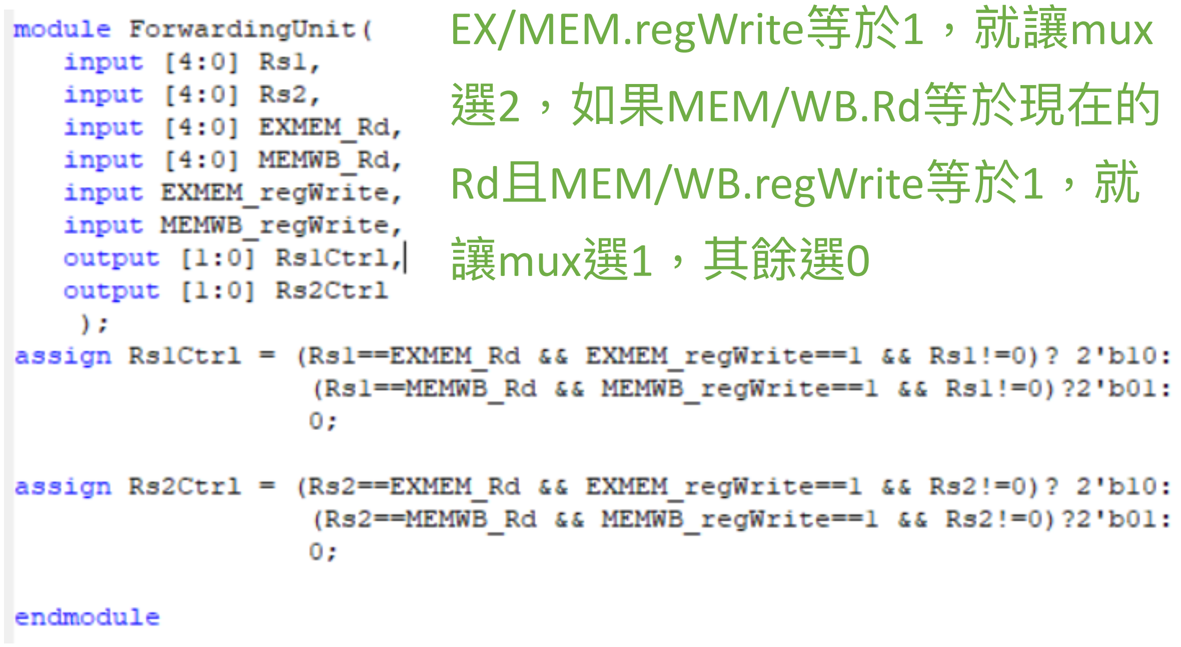
MEM/WB:



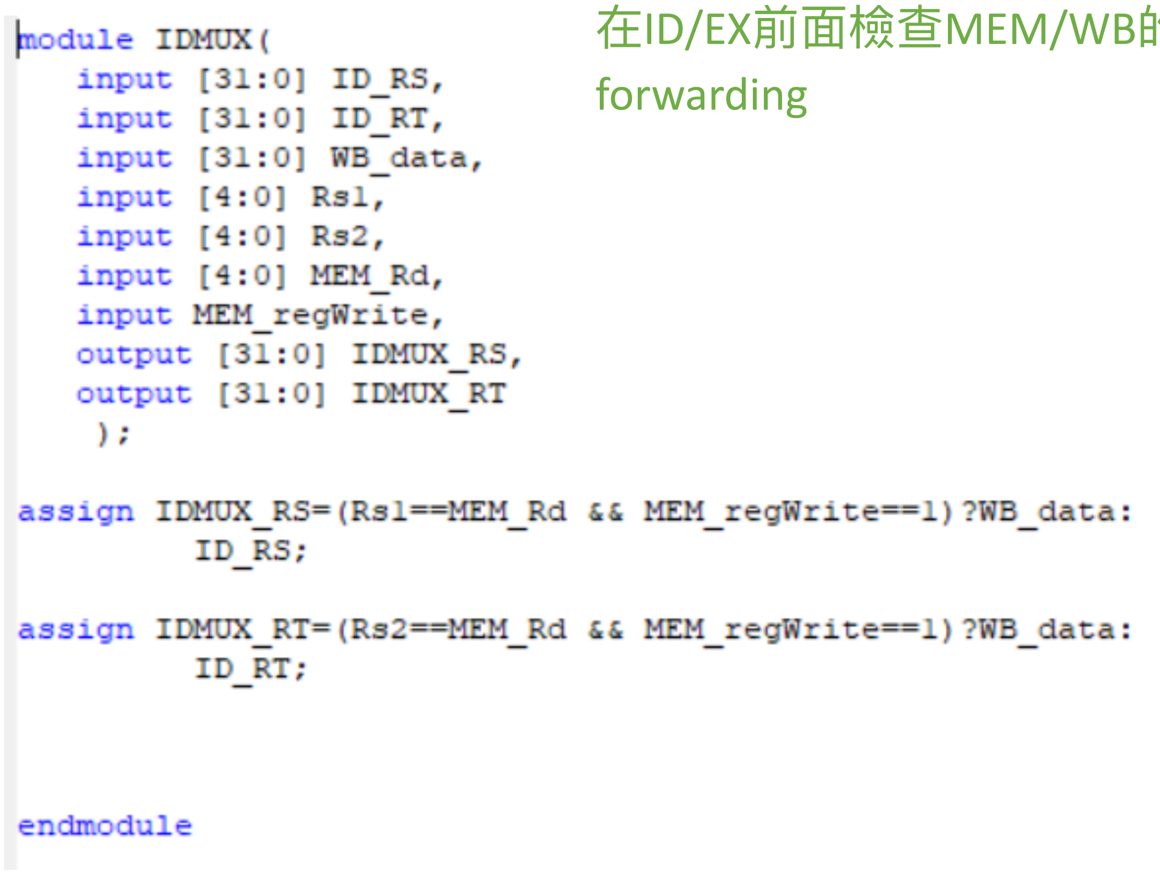
```
module MEMWB(
   input clk_i,
   input rst_i,
   input RegWrite_i,
   input [31:0] MemData_i,
   input [31:0] ALU_result_i,
   input [4:0] RDdata_i,
   input MemtoReg_i,
   output reg RegWrite_o,
   output reg MemtoReg_o,
   output reg [31:0] MemData_o,
   output reg [31:0] ALU_result_o,
   output reg [4:0] RDdata_o
    );
always@(posedge clk_i)begin
   if (rst_i) begin
      RegWrite_o=RegWrite_i;
      MemtoReg_o=MemtoReg_i;
      MemData_o=MemData_i;
      ALU_result_o=ALU_result_i;
      RDdata_o=RDdata_i;
   end
   else begin
      RegWrite_o=0;
      MemtoReg_o=0;
      MemData_o=0;
      ALU_result_o=0;
      RDdata_o=0;
   end
end

endmodule
```

Forwarding unit:

如果EX/MEM.Rd等於現在的Rd且EX/MEM.regWrite等於1，就讓mux選2，如果MEM/WB.Rd等於現在的Rd且MEM/WB.regWrite等於1，就讓mux選1，其餘選0



```
module ForwardingUnit(
   input [4:0] Rs1,
   input [4:0] Rs2,
   input [4:0] EXMEM_Rd,
   input [4:0] MEMWB_Rd,
   input EXMEM_regWrite,
   input MEMWB_regWrite,
   output [1:0] Rs1Ctrl,
   output [1:0] Rs2Ctrl
    );
assign Rs1Ctrl = (Rs1==EXMEM_Rd && EXMEM_regWrite==1 && Rs1!=0)? 2'b10:
                  (Rs1==MEMWB_Rd && MEMWB_regWrite==1 && Rs1!=0)?2'b01:
                  0;

assign Rs2Ctrl = (Rs2==EXMEM_Rd && EXMEM_regWrite==1 && Rs2!=0)? 2'b10:
                  (Rs2==MEMWB_Rd && MEMWB_regWrite==1 && Rs2!=0)?2'b01:
                  0;

endmodule
```

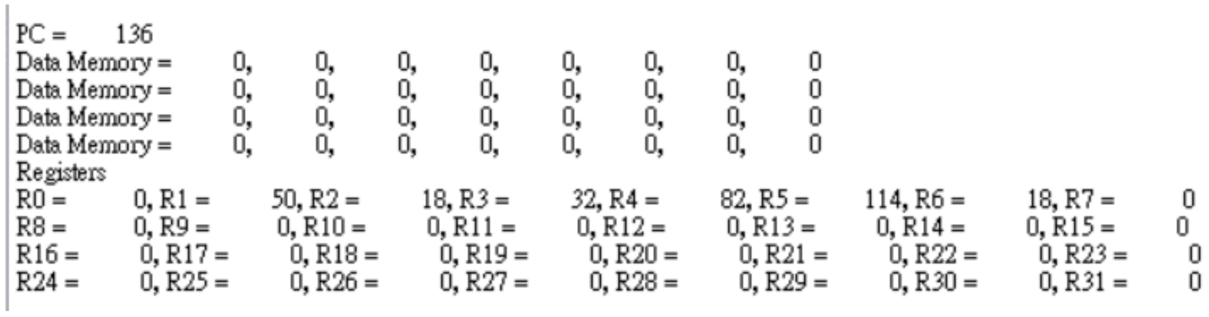


```
module IDMUX(
    input [31:0] ID_RS,
    input [31:0] ID_RT,
    input [31:0] WB_data,
    input [4:0] Rs1,
    input [4:0] Rs2,
    input [4:0] MEM_Rd,
    input MEM_regWrite,
    output [31:0] IDMUX_RS,
    output [31:0] IDMUX_RT
     );

assign IDMUX_RS=(Rs1==MEM_Rd && MEM_regWrite==1)?WB_data:
         ID_RS;

assign IDMUX_RT=(Rs2==MEM_Rd && MEM_regWrite==1)?WB_data:
         ID_RT;



endmodule
```

在ID/EX前面檢查MEM/WB的forwarding

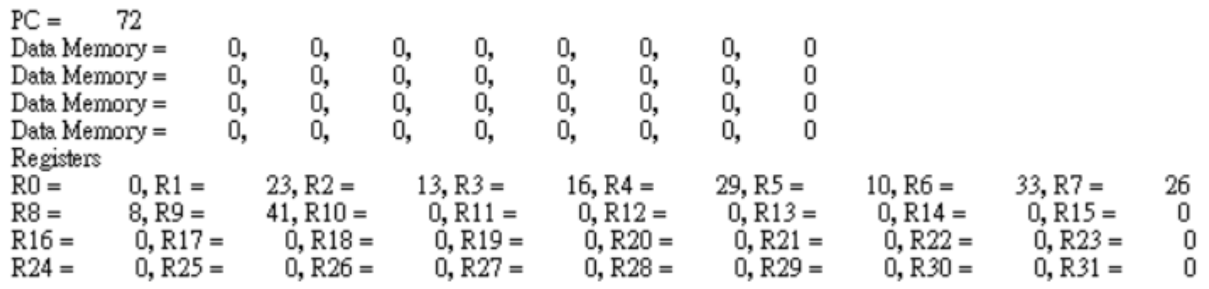
Implementation results:

Data 1 :



```
PC =      136
Data Memory =     0,      0,      0,      0,      0,      0,      0,      0
Data Memory =     0,      0,      0,      0,      0,      0,      0,      0
Data Memory =     0,      0,      0,      0,      0,      0,      0,      0
Data Memory =     0,      0,      0,      0,      0,      0,      0,      0
Registers
R0 =        0, R1 =       50, R2 =       18, R3 =       32, R4 =       82, R5 =      114, R6 =       18, R7 =        0
R8 =        0, R9 =        0, R10 =       0, R11 =       0, R12 =       0, R13 =       0, R14 =       0, R15 =       0
R16 =       0, R17 =       0, R18 =       0, R19 =       0, R20 =       0, R21 =       0, R22 =       0, R23 =       0
R24 =       0, R25 =       0, R26 =       0, R27 =       0, R28 =       0, R29 =       0, R30 =       0, R31 =       0
```

Data 2 :



```
PC =      72
Data Memory =     0,      0,      0,      0,      0,      0,      0,      0
Data Memory =     0,      0,      0,      0,      0,      0,      0,      0
Data Memory =     0,      0,      0,      0,      0,      0,      0,      0
Data Memory =     0,      0,      0,      0,      0,      0,      0,      0
Registers
R0 =        0, R1 =       23, R2 =       13, R3 =       16, R4 =       29, R5 =       10, R6 =       33, R7 =       26
R8 =        8, R9 =       41, R10 =       0, R11 =       0, R12 =       0, R13 =       0, R14 =       0, R15 =       0
R16 =       0, R17 =       0, R18 =       0, R19 =       0, R20 =       0, R21 =       0, R22 =       0, R23 =       0
R24 =       0, R25 =       0, R26 =       0, R27 =       0, R28 =       0, R29 =       0, R30 =       0, R31 =       0
```

Problems encountered and solutions:

原本按照課本給的圖，把forwarding unit寫在ID/EX後面，但後來發現Reg_file寫入會有一個cycle的延遲，所以按照hackmd上提供的方法，把MEM/WB的forwarding check寫在ID/EX前面。

Comment:

無