```
`timescale 1ns / 1ps

module CPU_Core(
    clk,
    reset,
//    Halt,

    //Branch + PC_ADD + Instruct. Input/Output (For Testing)
//    Branch_Result
//    Branch_Address,   <----Only uncomment when testing issues
regarding PC

    //Useful for identifying what the next PC address is and
current Instruction
//    Next_PC,
//    Current_Instr,

    //Controller Input/Output (For Testing)
//    ,RegWrite,
//    Immediate,
//    Mov,
//    BM,
    MemWrite,
    MemRead,
//    BranchCheck,
//    ALU,
//    MemReg,

    //RegFile Input/Output (For Testing)
//    ,RR_Write_Data,
//    RS1_Data,
//    RS2_Data,
//    RS1,
//    RS2,
//    Reg0_3_Data,
    RR_Data,
```

```
    //ALU Input/Output (For Testing)
//    ,ALU_Out,
    Concatenate

    //ALU-Mem Mux Input/Output (For Testing)
    ,DataMem
//    Data

    //Immediate Add Input/Output (For Testing)
//    ,Result

    //Immediate Mux Input/Output (For Testing)
//    ,WriteBack    <--- Only uncomment when testing RR Write
issues or IM Mux
    );
    input clk;
    input reset;
//    input Halt;

    //Branch + PC_ADD + Instruct. Input/Output (For Testing)
//    input wire Branch_Result;
//    input wire [15:0] Branch_Address; <----Only uncomment when
testing issues regarding PC

    //Useful for identifying what the next PC address is and
current Instruction
//    output [15:0] Next_PC;
//    output [15:0] Current_Instr;

    //Controller Input/Output (For Testing)
//    output RegWrite;
//    output Immediate;
//    output Mov;
//    output BM;
    output MemWrite;
    output MemRead;
//    output [2:0] BranchCheck;
```

```
//    output [2:0] ALU;
//    output MemReg;

    //RegFile Input/Output (For Testing)
//    input wire signed [15:0] RR_Write_Data;
//    output signed [15:0] RS1_Data;
//    output signed [15:0] RS2_Data;
//    output [3:0] RS1;
//    output [3:0] RS2;
//    output [15:0] Reg0_3_Data;
    output signed [15:0] RR_Data;

    //ALU Input/Output (For Testing)
//    output signed [15:0] ALU_Out;
    output signed [7:0] Concatenate;

    //ALU-Mem Mux Input/Output (For Testing)
    input signed [15:0] DataMem;
//    output signed [15:0] Data;

    //Immediate Add Input/Output (For Testing)
//    output signed [15:0] Result;

    //Immediate Mux Input/Output (For Testing)
//    output signed [15:0] WriteBack;     <--- Only uncomment when
testing RR Write issues or IM Mux
//    assign RR_Write_Data = WriteBack;   <--- Only uncomment when
testing RR Write issues or IM Mux

    //PC (ADD, Mux) + Instr. Mem Input/Output Wires
    wire [15:0] PC;
    wire [15:0] PC_Out;
    wire [15:0] IM_Out;
    wire [15:0] P_Add_Out;
    wire Halt;
    wire Branch_Result;
```

```
    //Controller Input/Output Wires
    wire RegWrite;
    wire Immediate;
    wire Mov;
    wire BM;
//    wire MemWrite;
//    wire MemRead;
    wire [2:0] BranchCheck;
    wire [2:0] ALU;
    wire MemReg;

    //Reg File Input/Output Wires
    wire signed [15:0] RR_Write_Data;
    wire signed [15:0] RS1_Data;
    wire signed [15:0] RS2_Data;
    wire [3:0] RS1;
    wire [3:0] RS2;
    wire[15:0] Reg0_3_Data;
//    wire signed [15:0] RR_Data;

    //ALU Input/Output Wires
    wire signed [15:0] ALU_Out;
//    wire signed [7:0] Concatenate;

    //ALU-Mem Mux Input/Output Wires
//    wire signed [15:0] DataMem;
    wire signed [15:0] Data;

    //Immediate Add Input/Output Wire
    wire signed [15:0] Result;

    //Useful for identifying what the next PC address is and
current Instruction
//    wire [15:0] Next_PC;
//    wire [15:0] Current_Instr;
//    assign Next_PC = PC;
//    assign Current_Instr = IM_Out;
```

```
    PC Call(.clk(clk), .reset(reset), .PC_In(PC), .Halt(Halt),
.PC_Out(PC_Out));
    Instruction_Mem Inst(.PC(PC_Out), .Instruct(IM_Out));
    PC_Add P_Add(.PC(PC_Out), .Result(P_Add_Out));

    //In order to test, the input 'Reg0_3_Data' must be replaced
with 'Branch_Address'
    PC_Mux P_Mux(.AND_Out(Branch_Result), .ADD(P_Add_Out),
.Branch(Reg0_3_Data), .PC_Change(PC));

    Controller Contr(.ControllerIn(IM_Out[3:0]), .reset(reset),
.RegWrite(RegWrite), .Immediate(Immediate), .Mov(Mov), .BM(BM),
.MemWrite(MemWrite), .MemRead(MemRead), .BranchCheck(BranchCheck),
.ALU(ALU), .MemReg(MemReg), .Halt(Halt));

    RegFile Reg(.clk(clk), .Instruct(IM_Out[15:4]), .mov(Mov),
.RegWrite(RegWrite), .BM(BM), .RR_Write_Data(RR_Write_Data),
.RS1_Data(RS1_Data), .RS2_Data(RS2_Data), .RS1(RS1), .RS2(RS2),
.Reg0_3(Reg0_3_Data), .Bit4_7(RR_Data));

    ALU Load(.RS1(RS1_Data), .RS1_Reg(RS1), .RS2(RS2_Data),
.RS2_Reg(RS2), .Funct(ALU), .Out(ALU_Out), .Concate(Concatenate));

    MemReg_Mux Mem_Mux(.MemReg(MemReg), .ALU(ALU_Out),
.DataMem(DataMem), .Data(Data));

    Immediate_Add IM_Add(.Data(Concatenate), .RegData(RR_Data),
.Result(Result));

    //In order to test, the output 'RR_Write_Data' must be replaced
with 'WriteBack'
    Immediate_Mux IM_Mux(.Immediate(Immediate), .Data(Data),
.ADD(Result), .RegData(RR_Write_Data));

    Branch_Check Check(.ALU_Out(ALU_Out), .Branch(BranchCheck),
.Result(Branch_Result));
```

```
    //Used for Testing DataMem before making Top-Top Level Design
    //Data_Mem Mem(.write_addr(RR_Data), .read_addr(Concatenate),
.RR_write_data(Concatenate), .memwrite_read(MemWrite_MemRead),
.reset(reset), .clk(clk), .read_data(DataMem));

endmodule
```