```
`timescale 1ns / 1ps

module PROCESSOR(

input clk,
    output reg [15:0]bus_core1,
    output reg [7:0]current_micro_instruction_core1,
    output reg [15:0]bus_core2,
    output reg [7:0]current_micro_instruction_core2,
    output reg [15:0]bus_core3,
    output reg [7:0]current_micro_instruction_core3,
    output reg [15:0]bus_core4,
    output reg [7:0]current_micro_instruction_core4,
    output reg [3:0]finish
    );
    wire [7:0]current_micro_instruction_wire_core1;
    wire [7:0]current_micro_instruction_wire_core2;
    wire [7:0]current_micro_instruction_wire_core3;
    wire [7:0]current_micro_instruction_wire_core4;
    wire [3:0]en_from_cpu_to_IM;
    wire [3:0]en_from_cpu_to_DM;
    wire [63:0]data_from_cpu_to_IM;
    wire [63:0]data_from_cpu_to_DM;
    wire [63:0]data_from_IM_to_cpu;
    wire [63:0]data_from_DM_to_cpu;
    wire [63:0]data_from_AR_to_M;
    wire [15:0]bus_wire_core1;
    wire [15:0]bus_wire_core2;
    wire [15:0]bus_wire_core3;
    wire [15:0]bus_wire_core4;
    wire [3:0]finish_wire;
    parameter proId0 =16'd0 ;
    parameter proId1 =16'd1 ;
    always @(*)
    begin

current_micro_instruction_core1<=current_micro_instruction_wire_core1;
        bus_core1 <= bus_wire_core1;

current_micro_instruction_core2<=current_micro_instruction_wire_core2;
        bus_core2 <= bus_wire_core2;

current_micro_instruction_core3<=current_micro_instruction_wire_core3;
        bus_core3 <= bus_wire_core3;

current_micro_instruction_core4<=current_micro_instruction_wire_core4;
        bus_core4 <= bus_wire_core4;
        finish<=finish_wire;
    end
    CORE core1 (
        .clk(clk),
        .data_DM_core(data_from_DM_to_cpu[15:0]),
        .data_IM_core(data_from_IM_to_cpu[15:0]),
        .proId(proId0),
        .data_core_IM(data_from_cpu_to_IM[15:0]),
```

```
        .data_core_DM(data_from_cpu_to_DM[15:0]),
        .DM_en(en_from_cpu_to_DM[0:0]),
        .IM_en(en_from_cpu_to_IM[0:0]),
        .AR_out(data_from_AR_to_M[15:0]),
        .finish(finish_wire[0:0]),
        .bus(bus_wire_core1),
        .current_micro_instruction(current_micro_instruction_wire_core1)
        );
    CORE core2 (
        .clk(clk),
        .data_DM_core(data_from_DM_to_cpu[31:16]),
        .data_IM_core(data_from_IM_to_cpu[31:16]),
        .proId(proId1),
        .data_core_IM(data_from_cpu_to_IM[31:16]),
        .data_core_DM(data_from_cpu_to_DM[31:16]),
        .DM_en(en_from_cpu_to_DM[1:1]),
        .IM_en(en_from_cpu_to_IM[1:1]),
        .AR_out(data_from_AR_to_M[31:16]),
        .finish(finish_wire[1:1]),
        .bus(bus_wire_core2),
        .current_micro_instruction(current_micro_instruction_wire_core2)
        );
    CORE core3 (
        .clk(clk),
        .data_DM_core(data_from_DM_to_cpu[47:32]),
        .data_IM_core(data_from_IM_to_cpu[47:32]),
        .proId(16'd2),
        .data_core_IM(data_from_cpu_to_IM[47:32]),
        .data_core_DM(data_from_cpu_to_DM[47:32]),
        .DM_en(en_from_cpu_to_DM[2:2]),
        .IM_en(en_from_cpu_to_IM[2:2]),
        .AR_out(data_from_AR_to_M[47:32]),
        .finish(finish_wire[2:2]),
        .bus(bus_wire_core3),
        .current_micro_instruction(current_micro_instruction_wire_core3)
        );
    CORE core4 (
        .clk(clk),
        .data_DM_core(data_from_DM_to_cpu[63:48]),
        .data_IM_core(data_from_IM_to_cpu[63:48]),
        .proId(16'd3),
        .data_core_IM(data_from_cpu_to_IM[63:48]),
        .data_core_DM(data_from_cpu_to_DM[63:48]),
        .DM_en(en_from_cpu_to_DM[3:3]),
        .IM_en(en_from_cpu_to_IM[3:3]),
        .AR_out(data_from_AR_to_M[63:48]),
        .finish(finish_wire[3:3]),
        .bus(bus_wire_core4),
        .current_micro_instruction(current_micro_instruction_wire_core4)
        );
    IRAM IRAM(
        .clk(clk),
        .write(en_from_cpu_to_IM),
        .address(data_from_AR_to_M),
```

```
106         .instr_in(data_from_cpu_to_IM),
107         .instr_out(data_from_IM_to_cpu)
108     );
109     DRAM DRAM (
110         .clk(clk),
111         .write(en_from_cpu_to_DM),
112         .address(data_from_AR_to_M),
113         .data_in(data_from_cpu_to_DM),
114         .data_out(data_from_DM_to_cpu)
115     );
116 endmodule
```