```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/26/2023 05:13:25 PM
// Design Name:
// Module Name: RegFile
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module RegFile(
    input clk,
    input [11:0] Instruct,
    input mov,
    input RegWrite,
    input BM,
    input signed [15:0] RR_Write_Data,
    output reg signed [15:0] RS1_Data,
    output reg signed [15:0] RS2_Data,
    output reg [3:0] RS1,
    output reg [3:0] RS2,
    output reg [15:0] Reg0_3,
    output reg signed [15:0] Bit4_7
```

```
    );

    reg [15:0] Register[0:15];
    reg [1:0] BMR;
    reg [3:0] RR;

    initial begin
    for (integer i = 0; i < 15; i=i+1) begin
            Register[i] = 0;
            BMR = 0;
            RR = 0;
            Register[15] = 0;
        end
    end

    always @ *
         begin

         BMR = Instruct [1:0];
         RR = Instruct[3:0];
         RS1 = Instruct[7:4];
         RS2 = Instruct[11:8];

         RS1_Data = Register[RS1]; //Data of RS1 Register
         RS2_Data = Register[RS2]; //Data of RS2 Register

         Reg0_3 = Register[BMR]; //Data of 2 LSBs of RR Register
for Branch Address

         Bit4_7 = Register[RR];  //Data of RR register for
Immediate Add

         Register[15] = 0;   //Constantly Assigns Register 15 to 0
so it can't be overwritten on accident
     end

     always @ (negedge clk) begin
```

```
        if (RegWrite == 1)
            Register[RR] = RR_Write_Data; //Data to be written back
into RR address
        if (mov == 1) begin
            Register[RR] = Register[RS1]; //Assigns RS1 Register
Data to Register RR
            Register[RS1] = Register[RS2]; //Assigns RS2 Register
Data to Register RS1
        end
        if (BM == 1) begin
            Register[BMR] = Instruct[11:2]; //Assigns bits 11-2 of
instruction to Register [BMR = RR]
        end
     end
endmodule
```