# 实验六源代码

**实验代码**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 14:19:56 05/07/2021

// Design Name:

// Module Name: mem

// Project Name:

// Target Devices:

// Tool versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module Mem(clk\_m,Mem\_Write,

R\_Addr\_A,R\_Addr\_B,W\_Addr,Write\_Reg,rst,clk,

ALU\_OP,offset,

Mem\_Addr,M\_R\_Data,R\_data\_A,R\_data\_B,ALU\_F,B);

input clk\_m,Mem\_Write;

input [4:0]R\_Addr\_A;//A端口读寄存器地址

input [4:0]R\_Addr\_B;//B端口读寄存器地址

input [4:0]W\_Addr;//写寄存器地址

input Write\_Reg;

input [2:0]ALU\_OP;

input rst,clk;

input [15:0]offset;

output [31:0]M\_R\_Data;

output [31:0]R\_data\_A;

output [31:0]R\_data\_B;

wire ZF,OF;

output [31:0]B;

output [31:0] ALU\_F;

output [7:0] Mem\_Addr;

add Data\_RAM (

.clka(clk\_m), // input clka

.wea(Mem\_Write), // input [0 : 0] wea

.addra(Mem\_Addr[7:2]), // input [5 : 0] addra

.dina(R\_data\_B), // input [31 : 0] dina

.douta(M\_R\_Data) // output [31 : 0] douta

);

assign B={{16{1'b0}},offset}; //扩展

REGS REGS\_1(R\_data\_A,R\_data\_B,M\_R\_Data,R\_Addr\_A,R\_Addr\_B,W\_Addr,Write\_Reg,rst,clk);

ALU ALU\_1(ALU\_OP,R\_data\_A,B,ALU\_F,ZF,OF);

assign Mem\_Addr=ALU\_F[7:0];

endmodule

module REGS(R\_Data\_A,R\_Data\_B,W\_Data,R\_Addr\_A,R\_Addr\_B,W\_Addr,Write\_Reg,rst,clk);

input clk;//写入时钟信号

input rst;//清零信号

input Write\_Reg;//写控制信号

input [4:0]R\_Addr\_A;//A端口读寄存器地址

input [4:0]R\_Addr\_B;//B端口读寄存器地址

input [4:0]W\_Addr;//写寄存器地址

input [31:0]W\_Data;//写入数据

output [31:0]R\_Data\_A;//A端口读出数据

output [31:0]R\_Data\_B;//B端口读出数据

integer i;

reg [31:0] REG\_Files[0:31];

initial

for(i=0;i<32;i=i+1) REG\_Files[i]<=0;

always@(posedge clk or posedge rst)

begin

if(rst)

for(i=0;i<32;i=i+1) REG\_Files[i]<=0;

else

if(Write\_Reg) REG\_Files[W\_Addr]<=W\_Data;

end

assign R\_Data\_A=REG\_Files[R\_Addr\_A];

assign R\_Data\_B=REG\_Files[R\_Addr\_B];

endmodule

module ALU(ALU\_OP,A,B,F,ZF,OF);

input [2:0] ALU\_OP;

input [31:0] A;

input [31:0] B;

output [31:0] F;

output ZF;

output OF;

reg [31:0] F;

reg C,ZF,OF;

always@(\*)

begin

C=0;

OF=0;

case(ALU\_OP)

3'b000:begin F=A&B; end

3'b001:begin F=A|B; end

3'b010:begin F=A^B; end

3'b011:begin F=~(A|B); end

3'b100:begin {C,F}=A+B;OF = A[31]^B[31]^F[31]^C; end

3'b101:begin {C,F}=A-B;OF = A[31]^B[31]^F[31]^C; end

3'b110:begin F=A<B; end

3'b111:begin F=B<<A; end

endcase

ZF = F==0;

end

endmodule