`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 2019/07/06 13:55:22

// Design Name: 王兴宇,刘凯鑫,胡耀宇,谭灿

// Module Name: data\_path

// Project Name: data\_path

// Target Devices: xc7a100tfgg484-2

// Tool Versions: Vivado 2019.1

// Description: 类TEC-8数据通路

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module data\_path

(

input QD, //手动脉冲

input ABUS,SBUS,MBUS, //总线开关

input S0,S1, //ALU操作

input RD1,RD0, //选择写入寄存器和读入A端口的寄存器

input RS0,RS1, //选择读入B端口的寄存器

input DRW,MEMW, //写入寄存器，内存开关

input LAR,LPC, //写入AR，PC开关

input SD7,SD6,SD5,SD4,SD3,SD2,SD1,SD0, //数据开关

input i\_clk, // 时钟信号输入，100MHZ

output wire [7:0]DBUS, //数据总线

output wire [7:0]ARBUS, //地址总线

output wire [7:0]PCBUS, //PC总线

output reg [6:0] keyboard\_val, // 七段显示器

output reg [7:0] keyboard // 八位七段显示器使能信号

);

reg LR3,LR2,LR1,LR0; //寄存器开关

reg [7:0]R0; //寄存器0

reg [7:0]R1; //寄存器1

reg [7:0]R2; //寄存器2

reg [7:0]R3; //寄存器3

reg [7:0]A; //端口A

reg [7:0]B; //端口B

reg [7:0]R; //结果寄存器

reg [7:0]DR; //数据总线寄存器

reg [7:0]AR; //地址总线寄存器

reg [7:0]PC; //PC寄存器

reg [7:0]MEM[7:0]; //8\*8内存空间

reg [17:0] cnt; // 8位7段显示刷新计数子

reg [3:0] flag; // 用来判断此时八位七段显示器刷新显示哪一个

//把总线寄存器和总线连接起来

assign DBUS = DR;

assign ARBUS = AR;

assign PCBUS = PC;

//时钟上升沿到来，记数子+1

always @ (posedge i\_clk)

begin

cnt <= cnt + 1'b1;

end

wire key\_clk = cnt[17]; //构造想要的固定频率脉冲信号

always @ (posedge key\_clk)

begin

flag <= (flag + 1'b1) % 4'd8; //判别8位7段显示哪位刷新

case(flag)

4'h0:begin

keyboard <= 8'b0111\_1111; //刷新第1个七段显示

case(A[7:4])

4'h0:keyboard\_val <= 7'b0000001; //显示0

4'h1:keyboard\_val <= 7'b1001111; //显示1

4'h2:keyboard\_val <= 7'b0010010; //显示2

4'h3:keyboard\_val <= 7'b0000110; //显示3

4'h4:keyboard\_val <= 7'b1001100; //显示4

4'h5:keyboard\_val <= 7'b0100100; //显示5

4'h6:keyboard\_val <= 7'b0100000; //显示6

4'h7:keyboard\_val <= 7'b0001111; //显示7

4'h8:keyboard\_val <= 7'b0000000; //显示8

4'h9:keyboard\_val <= 7'b0001100; //显示9

4'hA:keyboard\_val <= 7'b0001000; //显示A

4'hB:keyboard\_val <= 7'b1100000; //显示B

4'hC:keyboard\_val <= 7'b0001101; //显示C

4'hD:keyboard\_val <= 7'b1000010; //显示D

4'hE:keyboard\_val <= 7'b0110000; //显示E

4'hF:keyboard\_val <= 7'b0111000; //显示F

endcase

end

4'h1:begin

keyboard <= 8'b1011\_1111; //刷新第2个七段显示

case(A[3:0])

4'h0:keyboard\_val <= 7'b0000001; //显示0

4'h1:keyboard\_val <= 7'b1001111; //显示1

4'h2:keyboard\_val <= 7'b0010010; //显示2

4'h3:keyboard\_val <= 7'b0000110; //显示3

4'h4:keyboard\_val <= 7'b1001100; //显示4

4'h5:keyboard\_val <= 7'b0100100; //显示5

4'h6:keyboard\_val <= 7'b0100000; //显示6

4'h7:keyboard\_val <= 7'b0001111; //显示7

4'h8:keyboard\_val <= 7'b0000000; //显示8

4'h9:keyboard\_val <= 7'b0001100; //显示9

4'hA:keyboard\_val <= 7'b0001000; //显示A

4'hB:keyboard\_val <= 7'b1100000; //显示B

4'hC:keyboard\_val <= 7'b0001101; //显示C

4'hD:keyboard\_val <= 7'b1000010; //显示D

4'hE:keyboard\_val <= 7'b0110000; //显示E

4'hF:keyboard\_val <= 7'b0111000; //显示F

endcase

end

4'h2:begin

keyboard <= 8'b1101\_1111; //刷新第3个七段显示

case(B[7:4])

4'h0:keyboard\_val <= 7'b0000001; //显示0

4'h1:keyboard\_val <= 7'b1001111; //显示1

4'h2:keyboard\_val <= 7'b0010010; //显示2

4'h3:keyboard\_val <= 7'b0000110; //显示3

4'h4:keyboard\_val <= 7'b1001100; //显示4

4'h5:keyboard\_val <= 7'b0100100; //显示5

4'h6:keyboard\_val <= 7'b0100000; //显示6

4'h7:keyboard\_val <= 7'b0001111; //显示7

4'h8:keyboard\_val <= 7'b0000000; //显示8

4'h9:keyboard\_val <= 7'b0001100; //显示9

4'hA:keyboard\_val <= 7'b0001000; //显示A

4'hB:keyboard\_val <= 7'b1100000; //显示B

4'hC:keyboard\_val <= 7'b0001101; //显示C

4'hD:keyboard\_val <= 7'b1000010; //显示D

4'hE:keyboard\_val <= 7'b0110000; //显示E

4'hF:keyboard\_val <= 7'b0111000; //显示F

endcase

end

4'h3:begin

keyboard <= 8'b1110\_1111; //刷新第4个七段显示

case(B[3:0])

4'h0:keyboard\_val <= 7'b0000001; //显示0

4'h1:keyboard\_val <= 7'b1001111; //显示1

4'h2:keyboard\_val <= 7'b0010010; //显示2

4'h3:keyboard\_val <= 7'b0000110; //显示3

4'h4:keyboard\_val <= 7'b1001100; //显示4

4'h5:keyboard\_val <= 7'b0100100; //显示5

4'h6:keyboard\_val <= 7'b0100000; //显示6

4'h7:keyboard\_val <= 7'b0001111; //显示7

4'h8:keyboard\_val <= 7'b0000000; //显示8

4'h9:keyboard\_val <= 7'b0001100; //显示9

4'hA:keyboard\_val <= 7'b0001000; //显示A

4'hB:keyboard\_val <= 7'b1100000; //显示B

4'hC:keyboard\_val <= 7'b0001101; //显示C

4'hD:keyboard\_val <= 7'b1000010; //显示D

4'hE:keyboard\_val <= 7'b0110000; //显示E

4'hF:keyboard\_val <= 7'b0111000; //显示F

endcase

end

4'h4:begin

keyboard <= 8'b1111\_1111; //刷新第5个七段显示

end

4'h5:begin

keyboard <= 8'b1111\_1111; //刷新第6个七段显示

end

4'h6:begin

keyboard <= 8'b1111\_1101; //刷新第7个七段显示

case(R[7:4])

4'h0:keyboard\_val <= 7'b0000001; //显示0

4'h1:keyboard\_val <= 7'b1001111; //显示1

4'h2:keyboard\_val <= 7'b0010010; //显示2

4'h3:keyboard\_val <= 7'b0000110; //显示3

4'h4:keyboard\_val <= 7'b1001100; //显示4

4'h5:keyboard\_val <= 7'b0100100; //显示5

4'h6:keyboard\_val <= 7'b0100000; //显示6

4'h7:keyboard\_val <= 7'b0001111; //显示7

4'h8:keyboard\_val <= 7'b0000000; //显示8

4'h9:keyboard\_val <= 7'b0001100; //显示9

4'hA:keyboard\_val <= 7'b0001000; //显示A

4'hB:keyboard\_val <= 7'b1100000; //显示B

4'hC:keyboard\_val <= 7'b0001101; //显示C

4'hD:keyboard\_val <= 7'b1000010; //显示D

4'hE:keyboard\_val <= 7'b0110000; //显示E

4'hF:keyboard\_val <= 7'b0111000; //显示F

endcase

end

4'h7:begin

keyboard <= 8'b1111\_1110; //刷新第8个七段显示

case(R[3:0])

4'h0:keyboard\_val <= 7'b0000001; //显示0

4'h1:keyboard\_val <= 7'b1001111; //显示1

4'h2:keyboard\_val <= 7'b0010010; //显示2

4'h3:keyboard\_val <= 7'b0000110; //显示3

4'h4:keyboard\_val <= 7'b1001100; //显示4

4'h5:keyboard\_val <= 7'b0100100; //显示5

4'h6:keyboard\_val <= 7'b0100000; //显示6

4'h7:keyboard\_val <= 7'b0001111; //显示7

4'h8:keyboard\_val <= 7'b0000000; //显示8

4'h9:keyboard\_val <= 7'b0001100; //显示9

4'hA:keyboard\_val <= 7'b0001000; //显示A

4'hB:keyboard\_val <= 7'b1100000; //显示B

4'hC:keyboard\_val <= 7'b0001101; //显示C

4'hD:keyboard\_val <= 7'b1000010; //显示D

4'hE:keyboard\_val <= 7'b0110000; //显示E

4'hF:keyboard\_val <= 7'b0111000; //显示F

endcase

end

endcase

end

//根据总线开关判断哪个数据可以进入数据总线

always @(posedge QD)

begin

case({SBUS,ABUS,MBUS})

3'b100:DR = {SD7,SD6,SD5,SD4,SD3,SD2,SD1,SD0};

3'b010:DR = R;

3'b001:DR = MEM[AR];

default:DR=DR;

endcase

end

//2-4译码器选中寄存器

always @(\*)

begin

case({RD1,RD0})

2'b00:{LR3,LR2,LR1,LR0} = 4'b0001;

2'b01:{LR3,LR2,LR1,LR0} = 4'b0010;

2'b10:{LR3,LR2,LR1,LR0} = 4'b0100;

2'b11:{LR3,LR2,LR1,LR0} = 4'b1000;

endcase

end

always @(posedge QD)

begin

if(DRW)

case({LR3,LR2,LR1,LR0})

4'b0001:R0=DR;

4'b0010:R1=DR;

4'b0100:R2=DR;

4'b1000:R3=DR;

endcase

end

//判断哪个寄存器数据进入A，B端口

always @(\*)

begin

case({RD1,RD0})

2'b00:A=R0;

2'b01:A=R1;

2'b10:A=R2;

2'b11:A=R3;

endcase

end

always @(\*)

begin

case({RS1,RS0})

2'b00:B=R0;

2'b01:B=R1;

2'b10:B=R2;

2'b11:B=R3;

endcase

end

//判断ALU做哪种操作

always @(posedge QD)

begin

case({S1,S0})

2'b00:R=A;

2'b01:R=B;

2'b10:R=A+B;

2'b11:R=A-B;

endcase

end

//判断数据总线数据是否进入AR，PC

always @(posedge QD)

begin

case({LAR,LPC})

2'b01:PC=DR;

2'b10:AR=DR;

2'b11:begin

PC=DR;

AR=DR;

end

default:;

endcase

end

//判断数据总线数据是否存入内存相应单元

always @(posedge QD)

begin

if(MEMW)

MEM[AR]=DR;

end

endmodule