结合7条MIPS指令的CPU设计，讲解组合电路和时序电路的设计，使学生有目标、有兴趣。

仅供参考！！！

MIPS CPU的32位指令有三种类型

R 型、I 型、J 型

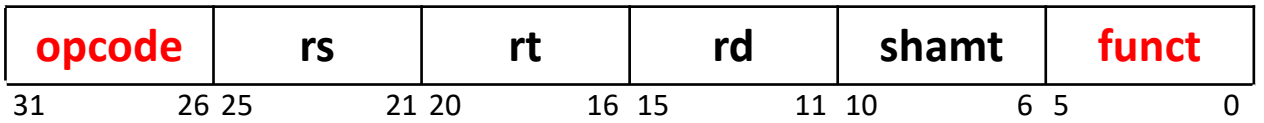
简单7条指令涉及

R型　addu（加法）
　　subu（减法）

I型　ori　（按位或立即数）
　　lui　（高位置立即数）
　　lw　（取字）
　　sw　（存字）
　　beq　（相等转移）

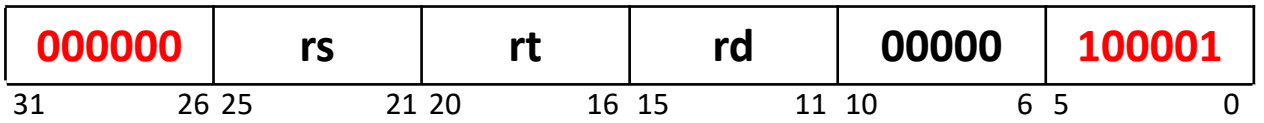
MIPS　　R类型指令格式

R类型



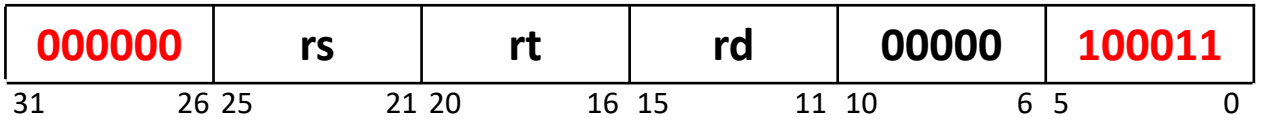
opcode恒为0，由funct决定R类型的具体指令

addu指令



加法指令　操作：R[rd]=R[rs]+R[rt]　　无溢出检测

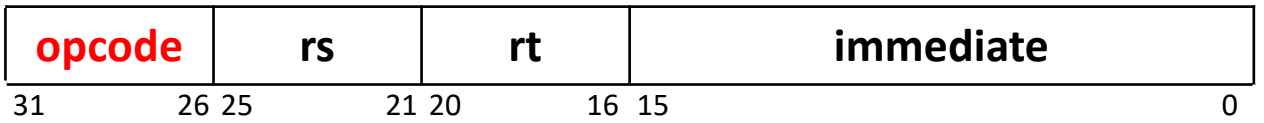
subu指令



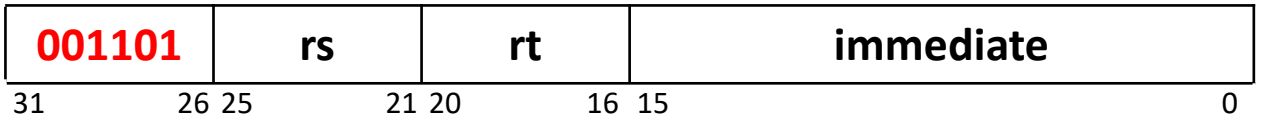
减法指令　操作：R[rd]=R[rs] - R[rt]　　无溢出检测

MIPS　　I类型指令格式

I类型



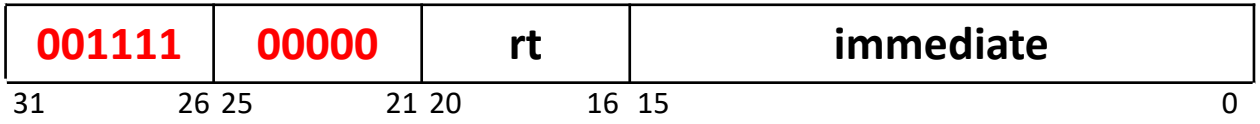
ori指令



按位或立即数指令　操作：R[rt]=R[rs]|ZeroExtImm

ZeroExtImm = {16{1b'0}, immediate}

lui指令

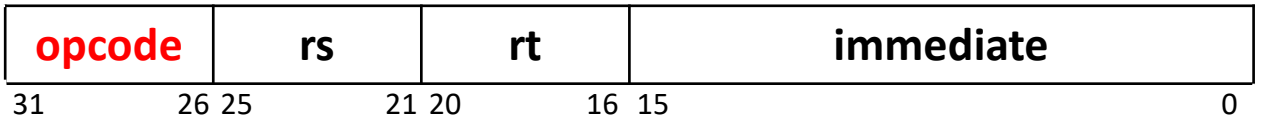


高位加载立即数指令　操作：R[rt]= {immediate,16'b0}

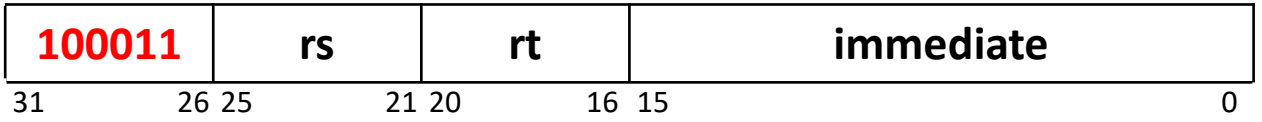
因约定R[0]恒为0，即：R[rt]= R[0] + {immediate,16'b0}

MIPS　　I类型指令格式

I类型



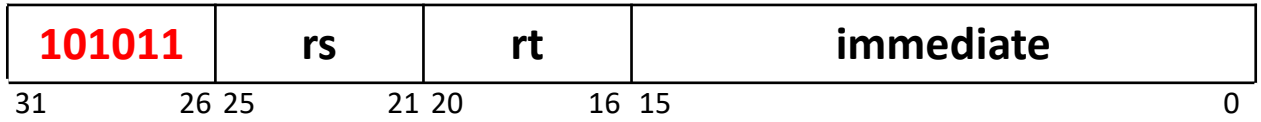
lw指令



取字指令　操作：R[rt]=M[R[s] + SignExtImm]

SignExtImm = { 16{immediate[15]} , immediate }

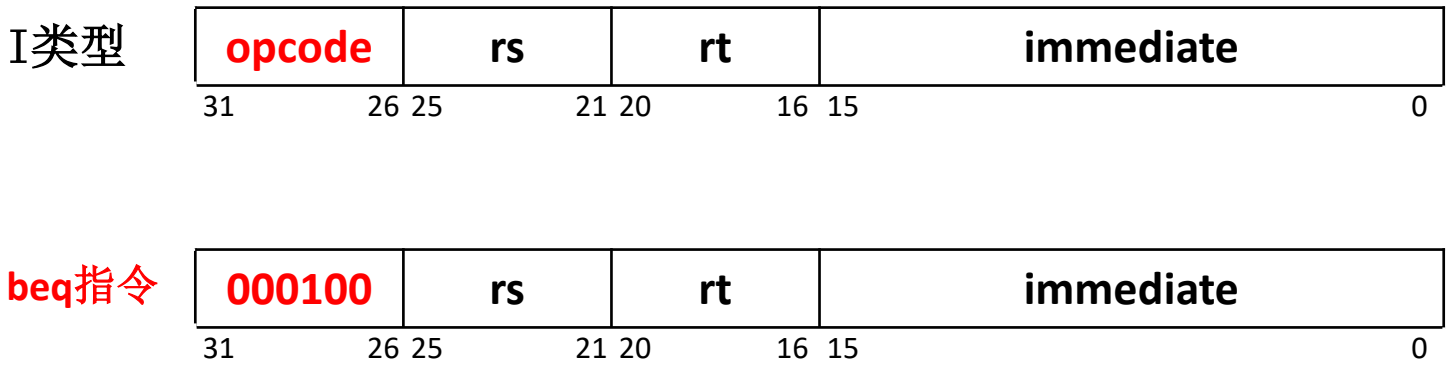
sw指令



存字指令　操作：M[R[rs] + SignExtImm] =R[rt]

SignExtImm = { 16{immediate[15]} , immediate }

MIPS　　I类型指令格式

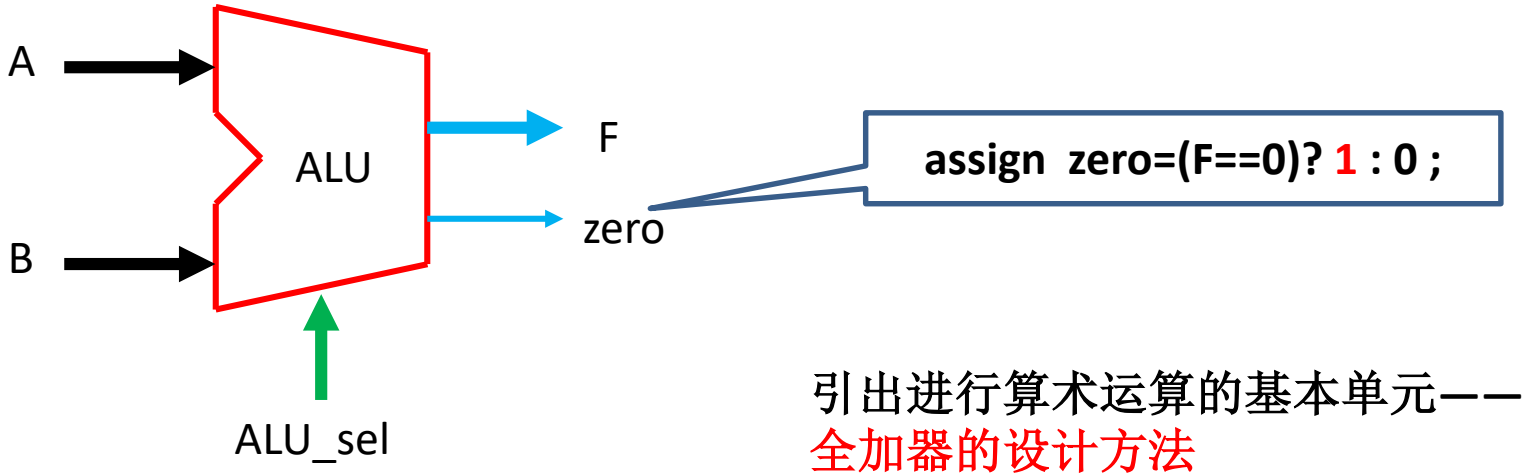


相等则跳转指令　　操作：if (R[rs]==R[rt])
PC=PC+4+BranchAddr

减法实现，如果R[rs] – R[rt] =0，则零标志Zero=1，否则Zero=0。

BranchAddr ={14{immediate[15]}, immediate,2'b0}

综合七条指令思考：应有一个能完成加、减、按位或运算的部件，即ALU



选择哪种运算结果输出到F
所以，ALU_Sel信号的位宽？

ALU_Sel[2:0]	F	
000	A+B	加
001	A-B	减
010	A|B	或
其他	0	预留

引出进行算术运算的基本单元——全加器的设计方法

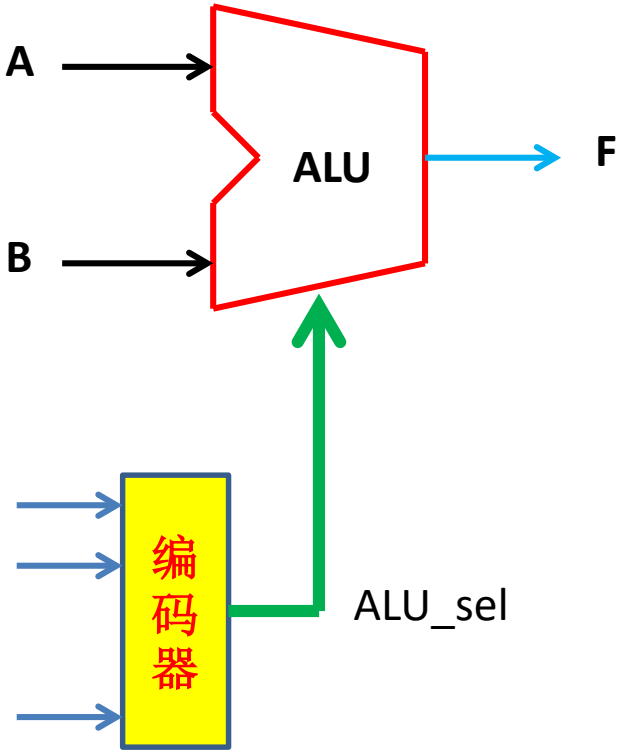
进而，讲解4位加法器的构成

然后，在LogicSim中构造一个能完成加、减、按位或三种运算的32位ALU并具有零标志Zero。

最后，介绍基于Verilog HDL的ALU设计模型 （32位!!!）

启发同学思考，如何增加新的运算功能？
例如：与运算、异或运算…

如何增加溢出的判断？

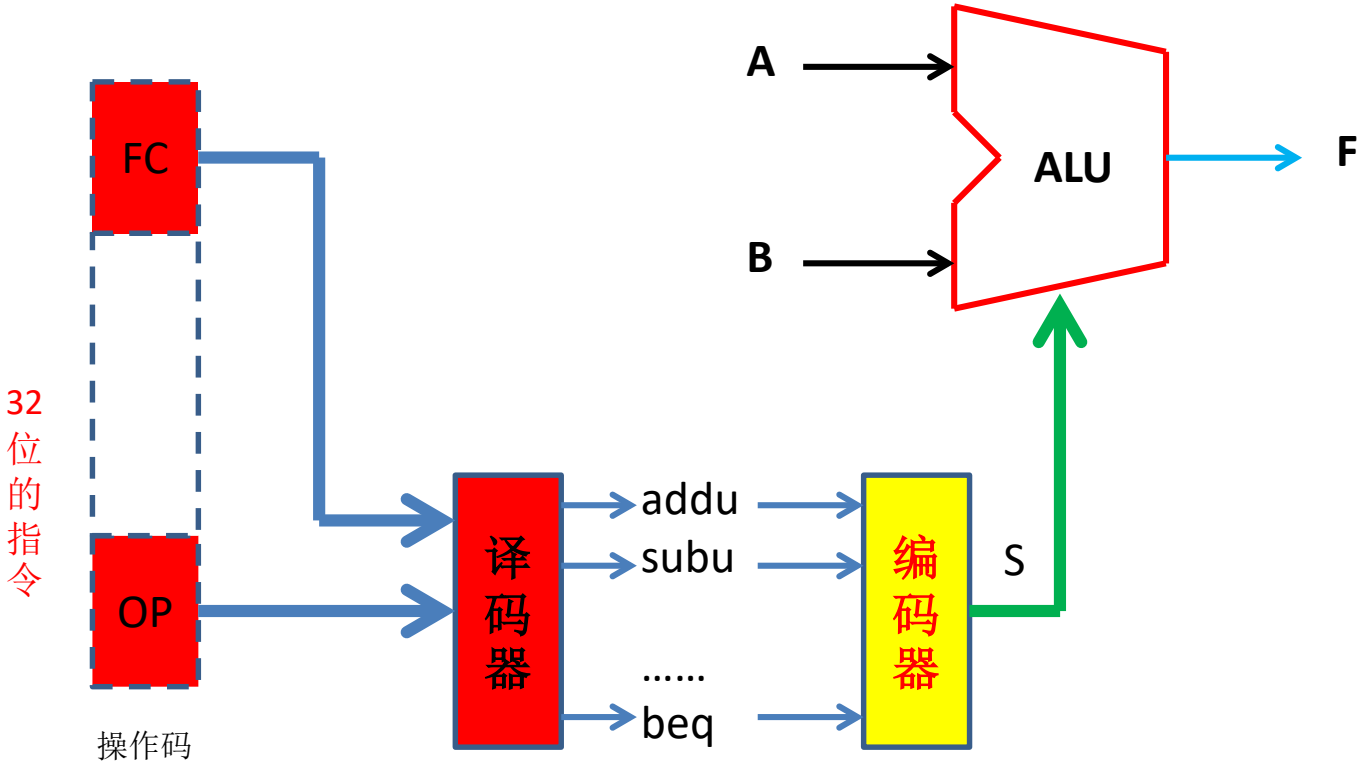


	ALU_S[2:0]
addu	000
subu	001
ori	010
lui	000
lw	000
sw	000
beq	001

介绍编码器的设计方法和HDL模型

引出优先权编码器的设计方法和HDL模型

七条指令ALU_sel编码器真值表（注意：概念引申）



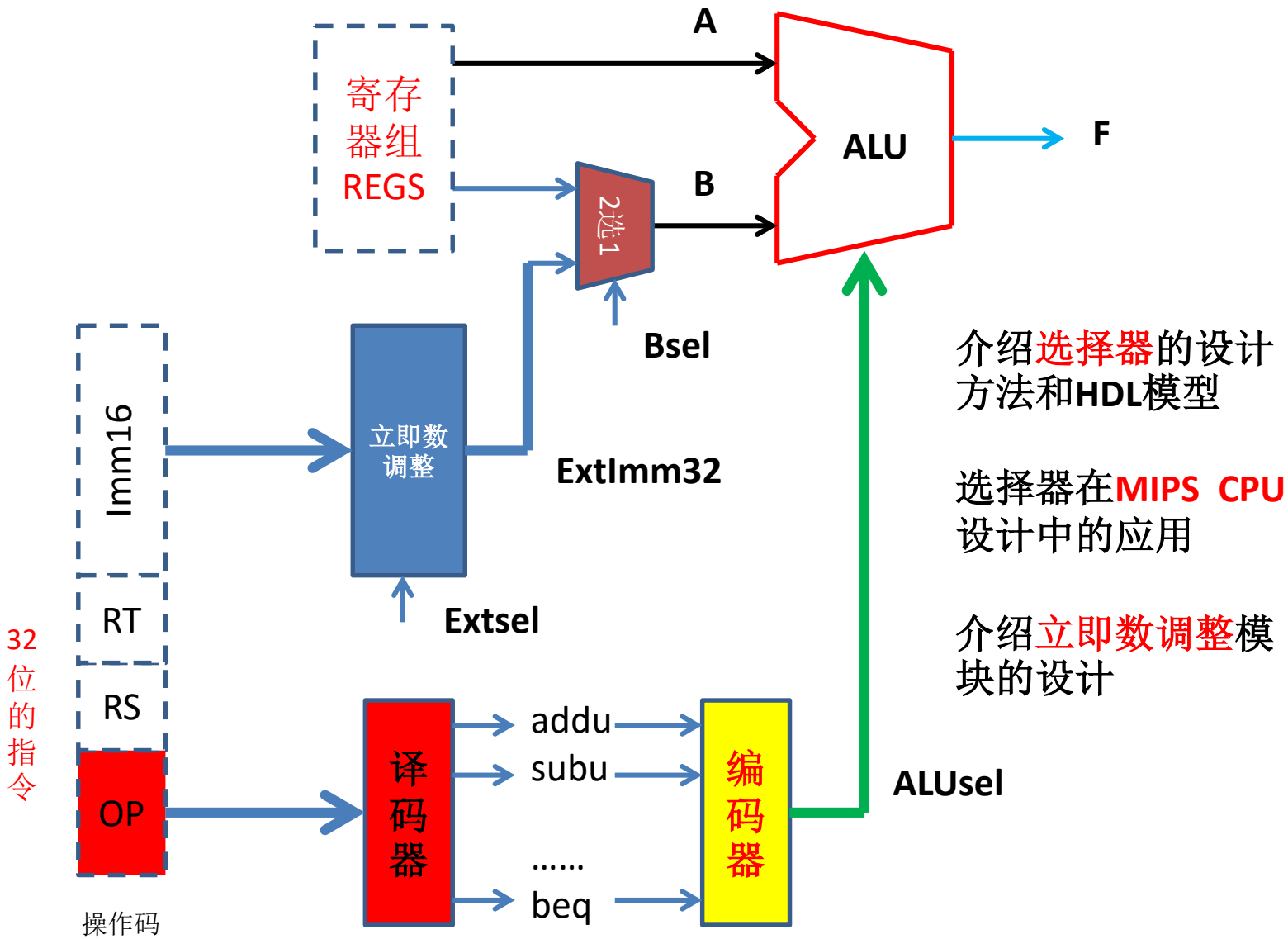
介绍译码器的设计方法和HDL模型

引出带使能端译码器的设计方法和HDL模型

7条MIPS指令译码---局部译码

输　入		输　出						
opcode	funct	addu	subu	ori	lui	lw	sw	beq
000000	100001	1	0	0	0	0	0	0
000000	100011	0	1	0	0	0	0	0
001101	x	0	0	1	0	0	0	0
001111	x	0	0	0	1	0	0	0
100011	x	0	0	0	0	1	0	0
101011	x	0	0	0	0	0	1	0
000100	x	0	0	0	0	0	0	1

基于LogicSim的电路实现？　　Vrilog HDL模型？



介绍选择器的设计方法和HDL模型

选择器在MIPS CPU设计中的应用

介绍立即数调整模块的设计

立即数扩展电路

在CPU的数据通路中有一个立即数扩展模块，其功能见下表，请用Verilog HDL建模。

信号名	位宽	方向	说明
Imm16	16	输入	来自指令寄存器的16位立即数
Extsel	2	输入	00：无符号扩展，将16位立即数进行0扩展至32位立即数； 01：符号扩展，将16位补码立即数扩展成32位补码立即数； 10：低位0扩展，将16位立即数移至32位立即数的高16位，低16位补0。
ExtImm32	32	输出	扩展后的32位立即数

MIPS指令 ori rt, rs, Imm16　　注释 R[rt] ←R[rs] | ZeroExtImm16

lui rt, Imm16　　R[rt] ← {Imm16, 16'b0}

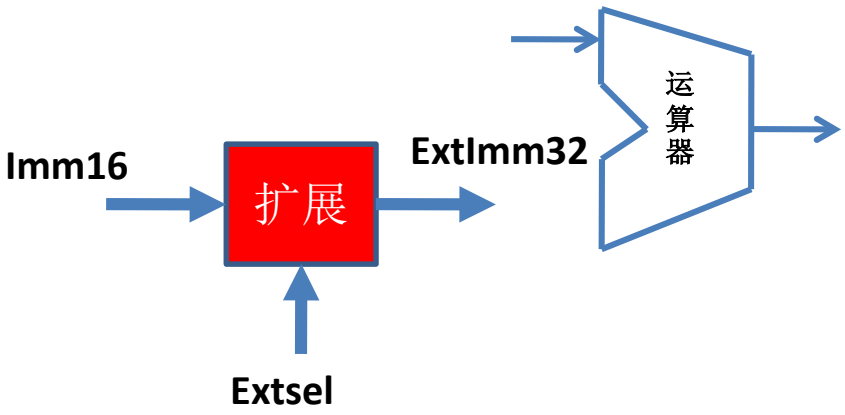
lw rt, rs, Imm16　　R[rt] ← M[R[rs] + SignExtImm]

sw rt, rs, Imm16　　M[R[rs] + SignExtImm] ← R[rt]

beq rs, rt, label　　if (R[rs]==R[rt])

PC=PC+4+BranchAddr （？另行处理）

CPU的运算器完成32位运算，
rt和rs是32位寄存器，
Imm16是16位，
所以需要将Imm16扩展成32位



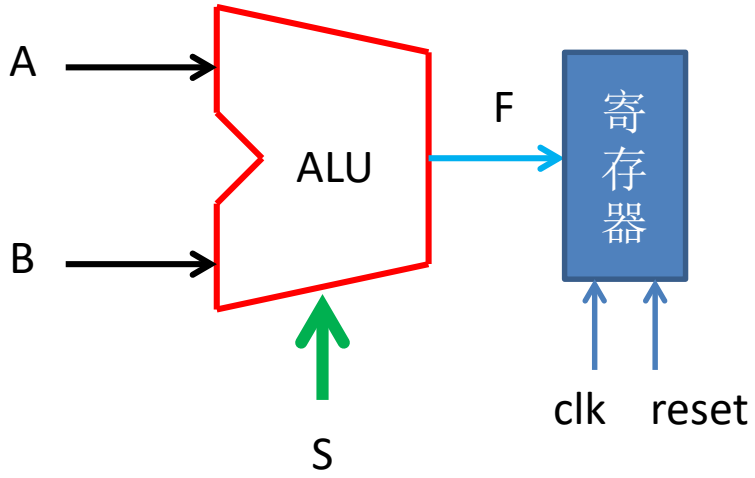
立即数扩展电路

基于LogicSim的实现？

Verilog HDL模型

```
module  ext_imm (imm16,extsel,imm32);
    input [15:0] imm16;
    input [1:0] extsel;
    output [31:0] imm32;
    reg  [31:0] imm32;
        always @(*)
            case  (extsel)
              0:  imm32=          //?无符号扩展     ori
              1:  imm32=          //?符号扩展   lw  sw  beq
              2:  imm32=          //?低位0扩展    lui
             default : imm32=       //?
          endcase
endmodule
```

同学补充完整



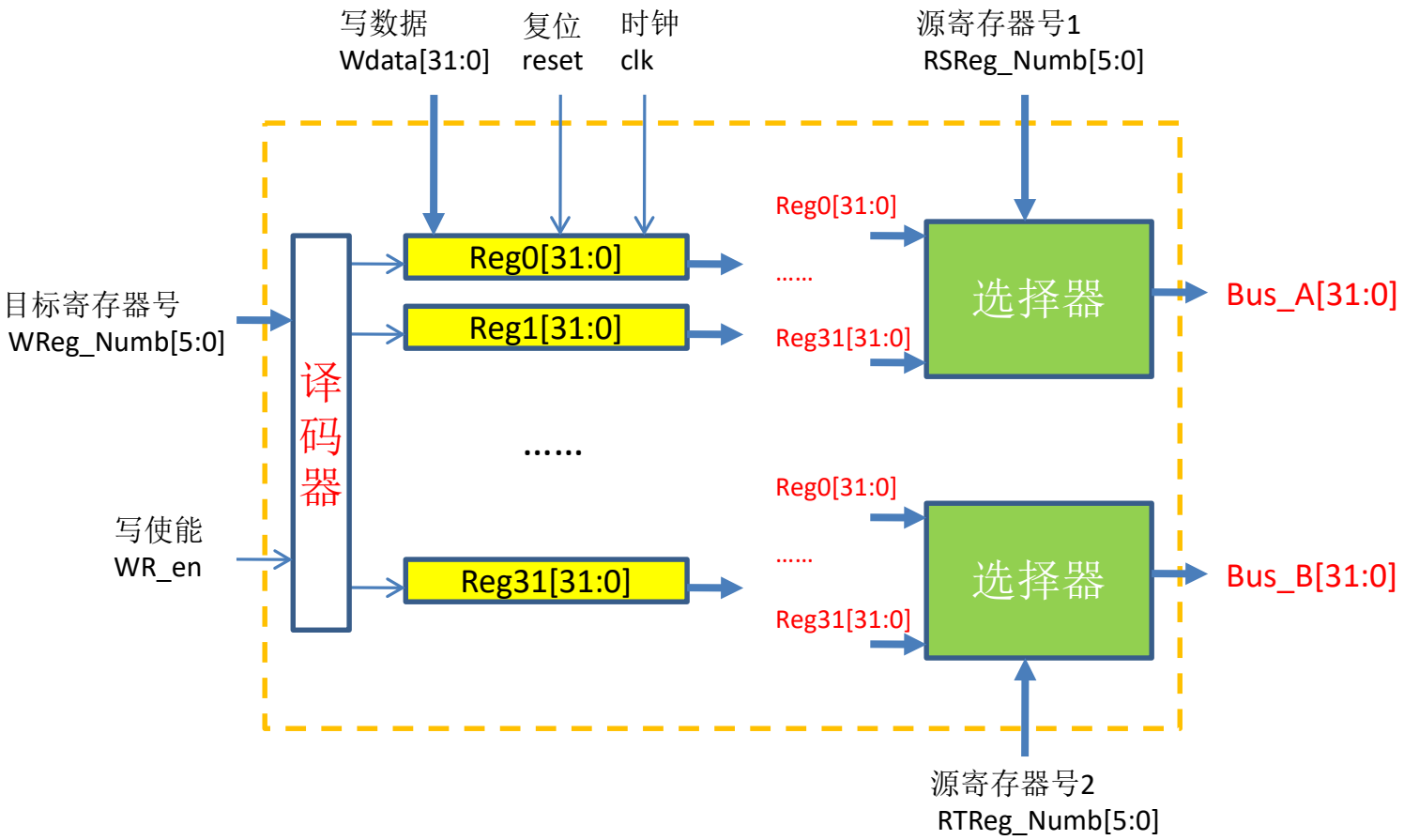
由运算结果需要保存，引出存储的概念、时序的概念

讲解双稳态元件-----RS、D、JK锁存器------D、JK触发器------寄存器

由指令中rs、rt、rd的定义引出，寄存器堆的概念

寄存器堆单元

MIPS指令 addu rd，rs，rt　解释 rd ← rs + rt



//在Verilog HDL 模块中，定义32个32位的寄存器堆

reg [31:0] Reg_files [31:0] ;

要求Reg0恒为0,如何处理?

寄存器堆单元

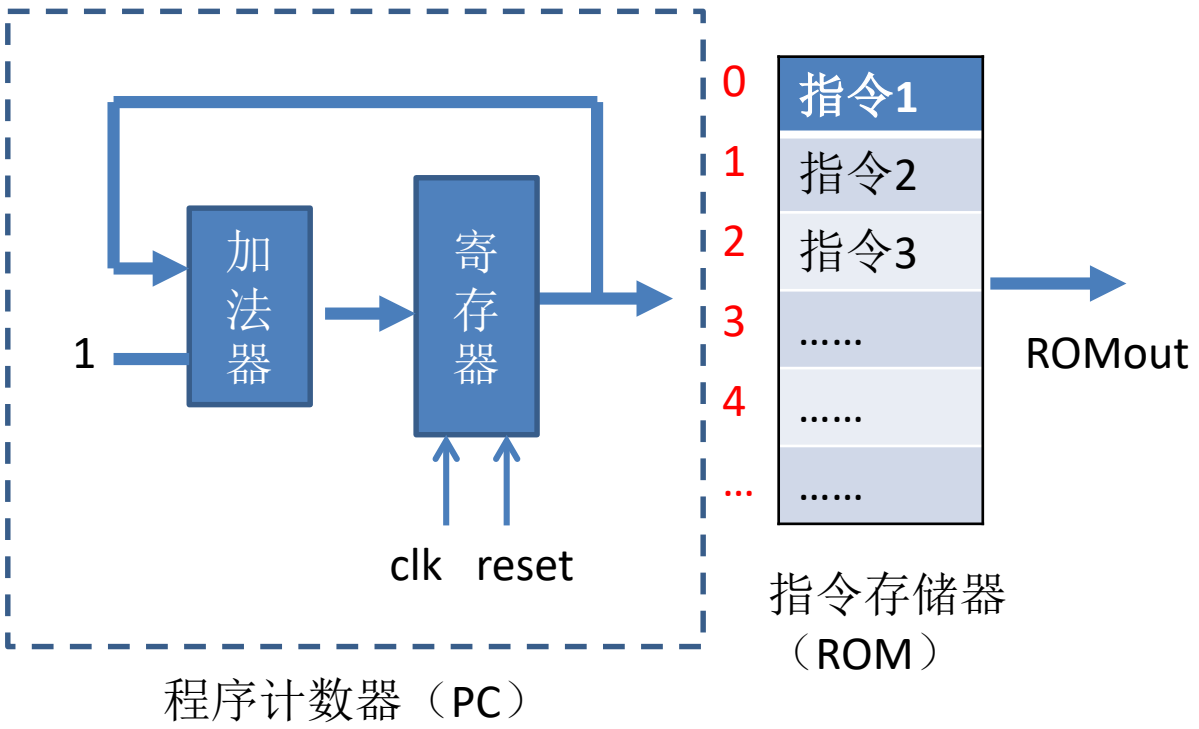
基于LogicSim的实现？

Verilog HDL模型

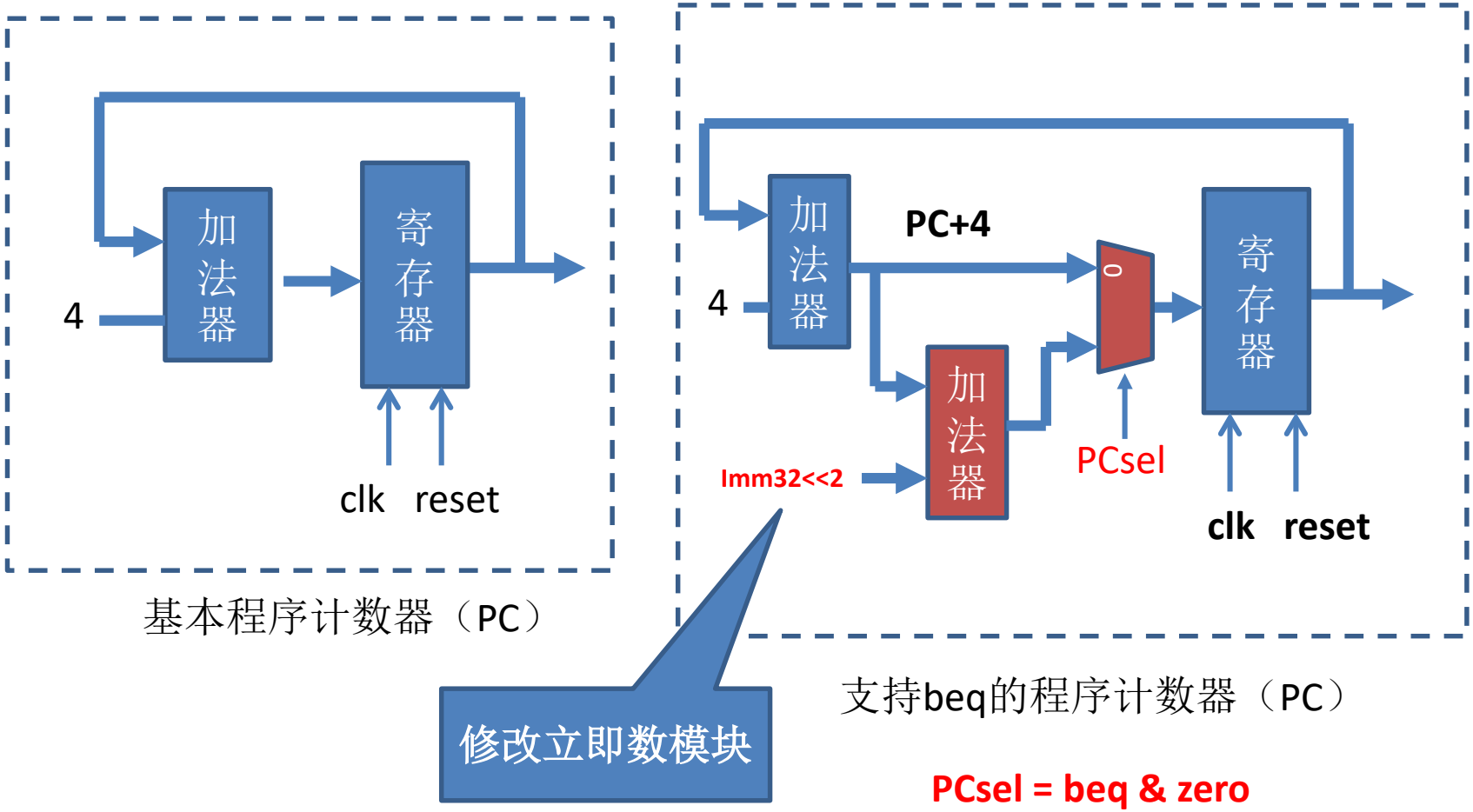
```
module   register (clk,reset,Wdata,Rwr,rd,rs,rt,rs_out,rt_out);
    input  clk,reset,Rwr;
    input  [31:0] Wdata;
    input  [4:0] rs,rt,rd;
    output  [31:0] rs_out,rt_out;
    reg  [31:0] Reg_files;
        assign  rs_out=reg_files[rs];
        assign  rt_out=reg_files[rt];
        always @(posedge  clk  or  posedge  reset)
          if  (reset==1)
                                        //清零
          else   if  (Rwr==1)
                    Reg_files[rd]=Wdata;
endmodule
```

同学补充完整

由取指顺序执行，引出指令存储器（ROM）和程序计数器（PC）



引出计数的概念------计数器的设计
（另一个文件---多种设计方法）



PCsel = beq & zero

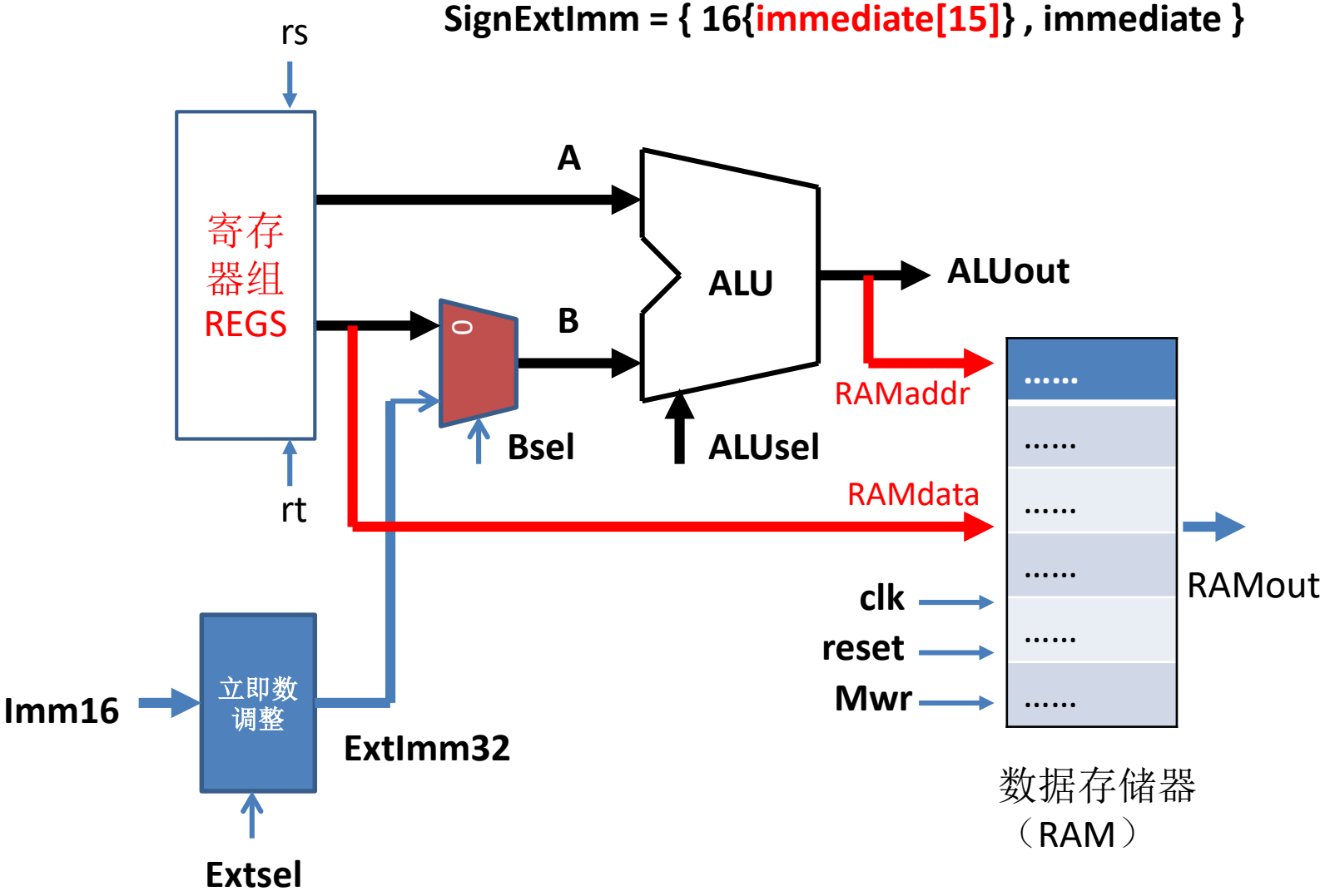
0：顺序执行

1：跳转

关于sw指令的实现

存字指令　操作：　M[R[rs] + SignExtImm] =R[rt]

SignExtImm = { 16{immediate[15]} , immediate }



在LogicSim中实现RAM？

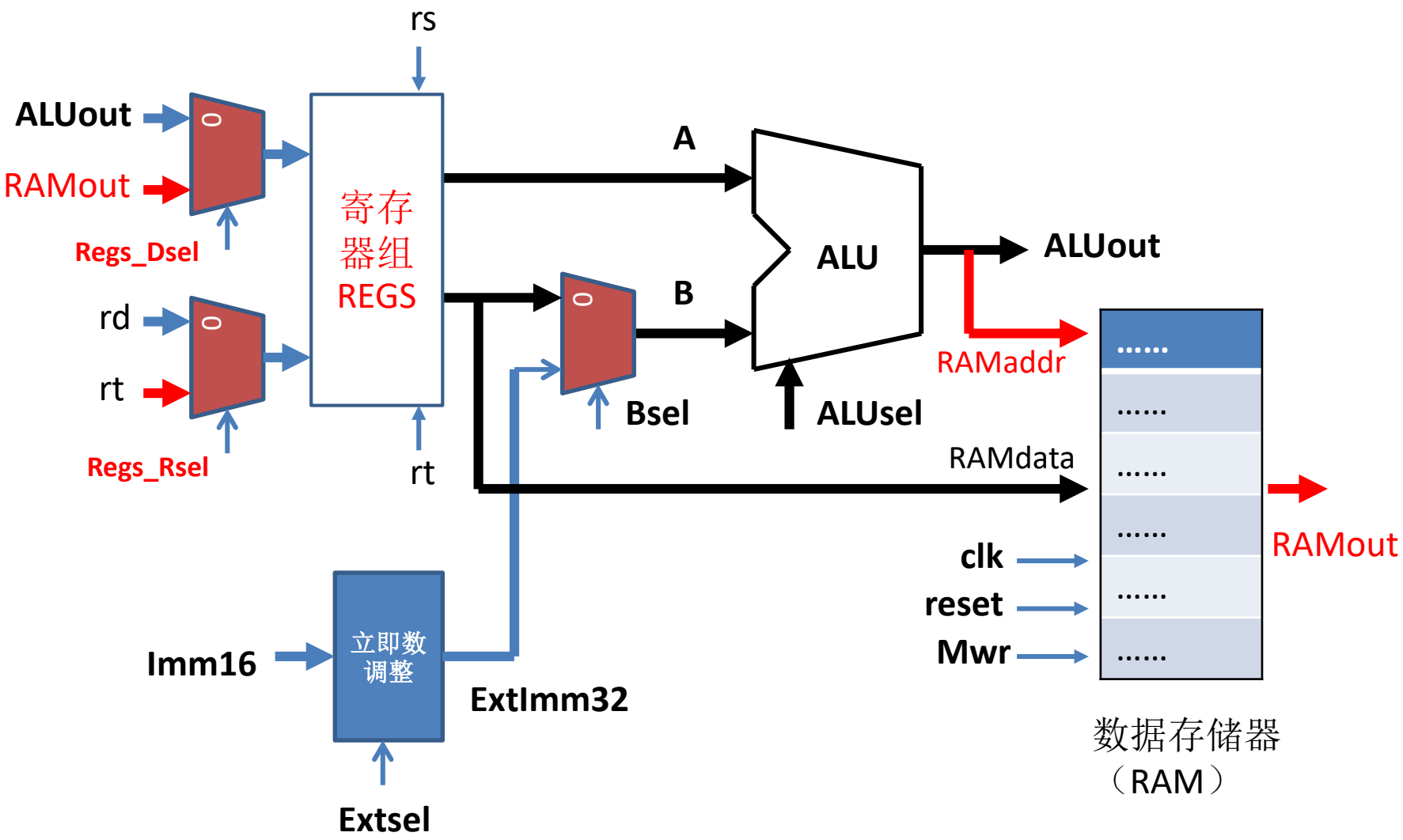
采用Verilog HDL实现RAM

```
module  RAM(clk,reset,Mwr,RAMdata,RAMaddr,RAMout);
    input  clk,reset,Mwr;
    input [31:0] RAMdata;
    input [7:0] RAMaddr;
    output   [31:0] RAMout;
    reg  [31:0] Memary [255:0];
        assign  RAMout= Memary [RAMaddr];  //?
        always @(posedeg  clk  or  posedge  reset)
            if  (reset)
                ……   //Memary清零
            else  if  (Mwr==1)
                    Memary [RAMaddr]=RAMdata;  //?
endmodule
```

关于lw指令的实现

取字指令　操作：　R[rt]=M[R[s] + SignExtImm]

SignExtImm = { 16{immediate[15]} , immediate }



根据addu、subu、ori、lui、sw、lw、beq讲解相关通路和控制信号

根据addu、subu、ori、lui、sw、lw、beq讲解相关通路和控制信号后，列表设计控制信号发生器。

指令译码器+控制信号发生器，称为控制器。

指令	ALU_sel [2:0]	EXT_sel [1:0]	B_sel	Regs_Dsel	Regs_Rsel	Reg_wr	M_wr	PC_sel
addu	000							0
subu	001							0
ori	010							0
lui	000							0
lw	000							0
sw	000							0
beq (zero=1)	001							1

ALU_sel[2]=0
ALU_sel[1]=ori
ALU_sel[0]=subu | beq

PC_sel=beq & zero

其它？？？同学完成

至此

基于7条MIPS指令的CPU设计思路和方法，已形成。

同学可进行LogicSim实现。

亦可以采用Verilog HDL实现。