**0.** **IorD:判断是在Memory中寻找指令[0]，还是寻找数据[1](数据的地址由指令计算得到)**

**1.** **MemRead:允许把Memory中的数据跟新到IR和MDR中[1],不然为[0]**

**2.** **MemWrite:允许把寄存器B中的数据写入Memory中[1]**

**3.** **IRwrite:允许把Memory中的数据跟新到IR中[1]**

**4.** **RegDst:选择IR的那几位代表将写入寄存器的地址15~11:[1] 20~16:[0]**

**5.** **MemtoReg:选择写入寄存器的数据来自Memory [1]还是来自运算结果[0]**

**6.** **RegWrite:允许将数据写入32个寄存器[1]**

**7. ALUsrcA:选择运算器的操作数1来自寄存器[1]还是来自PC[0]**

**8~9. ALUsrcB:选择运算器的操作数2来自**

**00:寄存器B**

**01: 4**

**10:IR中的立即数15~0**

**11:IR中的立即数\*4**

**10~11. ALUop:**

**00:op1+op2**

**01:op1-op2**

**10:由R指令中的操作符决定**

**11:**

**12~13. PCsource:**

**00：来自现在的计算结果**

**01：来自上一周期的运算结果**

**10：来自原PC与立即数的处理结果**

**11：无效信号**

**14. PCwrite:与PCwriteCond一起决定(or)是否允许数据写入PC**

**15. PCwriteCond:一般情况下为0，执行beq时为1**

**13个信号 16位**

**10个状态：由4位表示**

**S0:(所有)把指令与数据读入IR，MDR，让PC=PC+4**

**S1:(所有)寄存器Ａ，Ｂ读入应该读入的数据，ALUout读入PC与立即数相加后的数据，为BEQ，LW，SW做准备**

**S2:(L/S)将op1与立即数相加后存入ALUout**

**S3:(LW)将ALUout中的数据作为Memory地址将Memory的内容写入MDR**

**S4:(LW)将MDR中的数据根据S2计算的地址写入指定寄存器，进入S0**

**S5:(SW)将op2中的数据根据S2计算的Memory地址写入Memory，进入S0**

**S6: (R指令)将op1与op2运算之后的数据写入ALUout中**

**S7: (R指令)将ALUout中的数据写入指定的寄存器中，进入S0**

**S8: (beq指令)，计算两个操作数的相减若为0则将S1计算好的PC存入PC，不然不动，进入S0**

**S9: (J指令)，将PC与立即数相与后的数据存入PC，进入S0**

**0101000001000010**

**0000000011000000**

**0000000110000000**

**1100000110000000**

**0000011000000000**

**1010000110000000**

**0000000100100000**

**0000111000000000**

**0000000100010101**

**0000000000001010**

**在一个周期内sx发出自己应该发出的信号并根据操作符转化为下一个sy**

**S:0000**

**out 0101000001000010**

**---> 0001**

**S:0001**

**Out 0000000011000000**

**000010--->1001**

**000100--->1000**

**000000--->0110**

**100011--->0010**

**101011--->0010**

**S:0010**

**Out 0000000110000000**

**100011--->0011**

**101011--->0101**

**S:0011**

**Out 1100000110000000**

**--->0100**

**S:0100**

**Out 0000011000000000**

**--->0000**

**S:0101**

**Out 1010000110000000**

**--->0000**

**S:0110**

**Out 0000000100100000**

**--->0111**

**S:0111**

**Out 0000111000000000**

**--->0000**

**S:1000**

**Out 0000000100010101**

**--->0000**

**S:1001**

**Out 0000000000001010**

**--->0000**

**OP:**

**R:000000**

**LW:100011**

**SW:101011**

**BEQ:000100**

**J:000010**