|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Op | PCWre | ALUSrcA | ALUSrcB | DBDataSrc | InsMemRW | RegWre | /RD | /WR | RegDst | Ext  Sel |  |
| 000000(add) | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |  |
| 000001(addi) | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |  |
| 000010(sub) | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |  |
| 010000(ori) | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |  |
| 010001(and) | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |  |
| 010010(or) | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |  |
| 010011(andi) | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |  |
| 010100(xor) | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |  |
| 010101(xori) | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |  |
| 011000(sll) | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |  |
| 011100(slt) | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |  |
| 100110(sw) | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |  |
| 100111(lw) | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |  |
| 110000(beq) | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |  |
| 110001(bne) | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |  |
| 110010(bgtz) | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |  |
| 111000(j) | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |  |
| 111111(halt) | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |  |

|  |  |  |
| --- | --- | --- |
| Op | PCSrc | ALUOP |
| 000000(add) | 00 | 000 |
| 000001(addi) | 00 | 000 |
| 000010(sub) | 00 | 001 |
| 010000(ori) | 00 | 011 |
| 010001(and) | 00 | 100 |
| 010010(or) | 00 | 011 |
| 010011(andi) | 00 | 100 |
| 010100(xor) | 00 | 111 |
| 010101(xori) | 00 | 111 |
| 011000(sll) | 00 | 010 |
| 011100(slt) | 00 | 110 |
| 100110(sw) | 00 | 000 |
| 100111(lw) | 00 | 000 |
| 110000(beq) | 00(zero=0)  01(zero=1) | 111 |
| 110001(bne) | 01(zero=0)  00(zero=1) | 111 |
| 110010(bgtz) | 00(sign=1||zero=1)  01(sign=0&&zero=0) | 111 |
| 111000(j) | 10 | 000 |
| 111111(halt) | 00 | 000 |

**控制信号的作用**

|  |  |  |
| --- | --- | --- |
| 1 | 状态“0” | 状态“1” |
| **Reset** | 初始化PC为0 | PC接收新地址 |
| **PCWre** | PC不更改，相关指令：halt | PC更改，相关指令：除指令halt外 |
| **ALUSrcA** | 来自寄存器堆data1输出，相关指令：add、sub、addi、or、and、ori、beq、bne、bgtz、slt、sw、lw | 来自移位数sa，同时，进行(zero-extend)sa，即 {{27{0}},sa}，相关指令：sll |
| **ALUSrcB** | 来自寄存器堆data2输出，相关指令：add、sub、or、and、sll、slt、beq、bne、bgtz | 来自sign或zero扩展的立即数，相关指令：addi、ori、sw、lw |
| **DBDataSrc** | 来自ALU运算结果的输出，相关指令：add、addi、sub、ori、or、and、slt、sll | 来自数据存储器（Data MEM）的输出，相关指令：lw |
| **RegWre** | 无写寄存器组寄存器，相关指令：  beq、bne、bgtz、sw、halt、j | 寄存器组写使能，相关指令：add、addi、sub、ori、or、and、slt、sll、lw |
| **InsMemRW** | 写指令存储器 | 读指令存储器(Ins. Data) |
| **/RD** | 读数据存储器，相关指令：lw | 输出高阻态 |
| **/WR** | 写数据存储器，相关指令：sw | 无操作 |
| **RegDst** | 写寄存器组寄存器的地址，来自rt字段，相关指令：addi、ori、lw | 写寄存器组寄存器的地址，来自rd字段，相关指令：add、sub、and、or、slt、sll |
| **ExtSel** | (zero-extend)**immediate（**0扩展**），**相关指令：ori | (sign-extend)**immediate（**符号扩展**）**  ，相关指令：addi、sw、lw、bne、bne、bgtz |
| **PCSrc[1..0]** | 00：pc<－pc+4，相关指令：add、addi、sub、or、ori、and、slt、  sll、sw、lw、beq(zero=0)、bne(zero=1)、bgtz(sign=1，或zero=1)；  01：pc<－pc+4+(sign-extend)**immediate**，相关指令：beq(zero=1)、  bne(zero=0)、bgtz(sign=0，zero=0)；  10：pc<－{(pc+4)[31..28],addr[27..2],0,0}，相关指令：j；  11：未用 | |
| **ALUOp[2..0]** | ALU 8种运算功能选择(000-111)，看功能表 | |

**Control Unit：**

Input: sign、zero（来自ALU的输出）、Op（来自指令储存器）

Output:PCWre、ALUSrcA、ALUSrcB、**DBDataSrc、RegWre、/RD、/WR、RegDst、ExtSel、PCSrc[1:0]、ALUOP[2:0]**

**ALU运算功能表**

|  |  |  |
| --- | --- | --- |
| **ALUOp[2..0]** | **功能** | **描述** |
| **000** | **Y = A + B** | **加** |
| **001** | **Y = A – B** | **减** |
| **010** | **Y = B<<A** | **B左移A位** |
| **011** | **Y = A ∨ B** | **或** |
| **100** | **Y = A ∧ B** | **与** |
| **101** | **Y=（A<B）?1: 0** | **比较A与B**  **不带符号** |
| **110** | **if (A<B &&(A[31] == B[31] ))**  **Y = 1;**  **else if ( A[31] && !B[31) Y = 1;**  **else Y = 0;** | **比较A与B**  **带符号** |
| **111** | **Y = A ⊕ B** | **异或** |

zero，运算结果标志，**结果为0**，则zero=1；否则zero=0

sign，运算结果标志，**结果最高位为0**，则sign=0，正数；否则，sign=1，负数

ALU的输入：

数据输入：ReadData1(寄存器的输出),ReadData2(寄存器的另一个输出),sa,inExt（来自extend）

控制输入：ALUSrcA、ALUSrcB（来自控制单元）、ALUOp

|  |  |  |  |
| --- | --- | --- | --- |
| ALUSrcA | ALUSrcB | A | B |
| 0 | 0 | ReadData1 | ReadData2 |
| 0 | 1 | ReadData1 | inExt |
| 1 | 0 | sa | ReadData2 |
| 1 | 1 | sa | inExt |

PC：

Input:PrcWre(控制单元)、PrcSrc(控制单元)、immediate、CLK

Output:Address

|  |  |  |  |
| --- | --- | --- | --- |
| PrcWre | PrcSrc | Reset | Address(次态) |
| x | x | 0 | 0 |
| 0 | x | 1 | 不变 |
| 1 | 00 | 1 | Address+4 |
| 1 | 01 | 1 | Address+4+(sign-extend)immediate |
| 1 | 10 | 1 | {(Address+4)[31..28],addr[27..2],0,0} |
| 1 | 11 | 1 | 不变 |

SignZeroExtend:

|  |  |  |
| --- | --- | --- |
| ExtSel | Sign | 扩展方式 |
| 0 | x | 前16位补0 |
| 1 | 0 | 前16位补0 |
| 1 | 1 | 前16位补1 |

**DataMemory：**

**Input:DAddr (ALU的Result输出)、CLK、RD（来自控制单元）、WR(来自控制单元)、DataIn(来自寄存器的ReadData2)**

|  |  |  |  |
| --- | --- | --- | --- |
| /RD | /WR | DataOut | 功能 |
| 0 | 1 | Memory[DAddr] | DataOut= Memory[DAddr] |
| 1 | 0 | 1 | Memory[DAddr]=DataIn |
| 1 | 1 | 1 | 无 |

**instructionMemory:**

Input:InsMemRW（从控制单元）、IDataIn(用于写指令)、Iaddr(PC的Address输出)

Output:DataOut(op、rs、rt、rd、sa、immediate)

PC的Address输出为地址，总共有20条指令，则Iaddr[5:2]的范围为0~19，(在PC中每一次地址加4即相当于Iaddr[2]加1)。

在指令储存器中有Memory（二维数组）专门负责储存指令的16进制数值，Memory[Iaddr[5:2]]就是得到当前地址的指令值。

对于每一个Memory[i],

Op=Memory[i][31:26]，

Rs=Memory[i][25:21],

rt=Memory[i][20:16],

rd=Memory[i][15:11],

sa= Memory[i][10:6]（仅仅用于左移），

immediate=Memory[i][15:0](用于有立即数的运算),

address=Memory[i][25:0](仅用于j、halt)

**registerFile：**

**Input:DBDataSrc(来自控制单元)、ReadReg1(来自指令储存器的rs)、ReadReg2(来自指令储存器的rt)、RegDst(来自控制单元)、DataOutFromData(来自数据存储器的DataOut)、DataOutFromALU(来自ALU的result)、rd(来自指令储存器的rd)、WE(来自控制单元的RegWre)、CLK**

**Output:ReadData1、ReadData2**

**WriteReg=(RegDst==0)?rt:rd**

**(WriteReg是将数据写入的寄存器端口，当RegDst为0时rt写入该端口，否则rd写入)**

**WriteData=(DBDataSrc==0)?DataOutFromALU:DataOutFromData**

**(WriteData是要写入寄存器的数据)**

**Register是寄存器内储存数据的变量**

**ReadData1=Register[ReadReg1],ReadData2=Register[ReadReg2]**

当WE==0时，不需要写数据

当WE==1时，Register[WriteReg]=WriteData，注意要判断WriteReg是否为0，因为不可以对0号寄存器写入数据

**singleStyleCPU（顶层模块）：**

ALU->PC->ControlUnit->DataMemory->

Instruction Memory->RegistFile->signZeroExtend

测试模块：

Reset=1

#100 clk=~clk