**（实验二）单周期CPU数据通路图与控制信号**

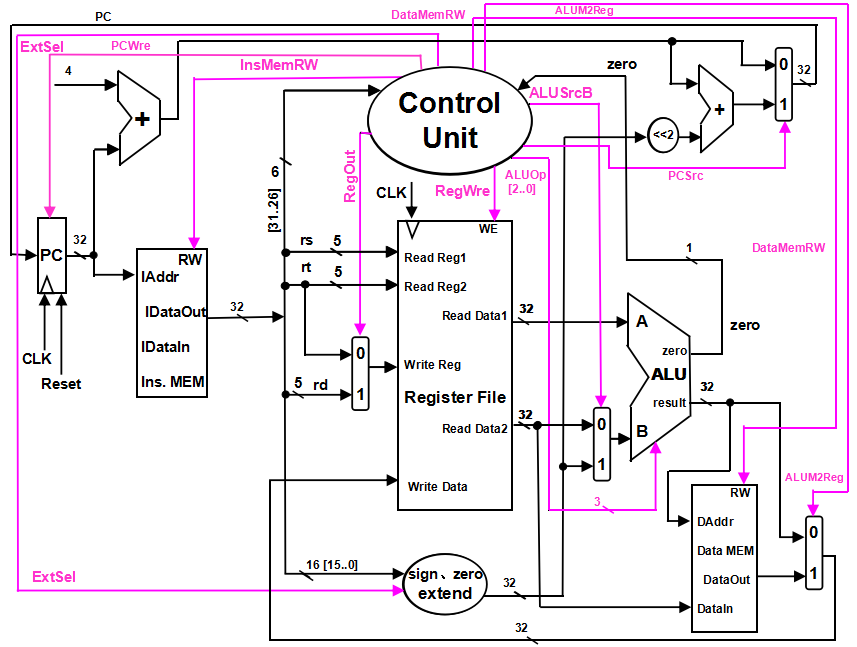


表1是根据指令的功能和以上数据通路图的关系而得出来的控制信号的作用一览表。搞清楚这张表的关系，有利于生成“表3 控制信号与指令的关系表”，和最后生成控制信号的逻辑表达式。上图有关引脚说明，请看“单周期CPU设计实验（简单型）.DOC”文档。

**表1 控制信号的作用**

|  |  |  |
| --- | --- | --- |
| 控制信号名 | 状态“0” | 状态“1” |
| PCWre | PC不更改，相关指令：halt | PC更改，相关指令：除指令halt外 |
| ALUSrcBB | 来自寄存器堆data2输出，相关指令：add、sub、or、and、move、beq | 来自sign或zero扩展的立即数，相关指令：addi、ori、sw、lw |
| ALUM2Reg | 来自ALU运算结果的输出，相关指令：add、addi、sub、ori、or、and、move | 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：  sw、halt | 寄存器组写使能，相关指令：add、addi、sub、ori、or、and、move、lw |
| InsMemRW | 读指令存储器(Ins. Data)，初始化为0 | 写指令存储器 |
| DataMemRW | 读数据存储器，相关指令：lw | 写数据存储器，相关指令：sw |
| ExtSel | 相关指令：ori，(zero-extend)**immediate（**0扩展**）** | 相关指令：addi、sw、lw、beq，  (sign-extend)**immediate（**符号扩展**）** |
| PCSrc | PC←PC+4，相关指令：add、sub、ori、or、and、move、sw、lw、beq(zero=0) | PC←PC+4+(sign-extend)**immediate**，同时zero=1，相关指令：beq |
| RegOut | 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、move |
| ALUOp[2..0] | ALU 8种运算功能选择(000-111)，看功能表 | |

**表2 ALU运算功能表**

|  |  |  |
| --- | --- | --- |
| **ALUOp[2..0]** | **功能** | **描述** |
| **000** | **A + B** | **加** |
| **001** | **A – B** | **减** |
| **010** | **B – A** | **减** |
| **011** | **A ∨ B** | **或** |
| **100** | **A ∧ B** | **与** |
| **101** | **/A ∧ B** | **A非与B** |
| **110** | **A ⊕ B** | **异或** |
| **111** | **A ⊙ B** | **同或** |

**表3 控制信号与指令的关系表**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **控制信号** | | | | | | | | | | |
| **指令** | **z** | **PCWre** | **ALUSrcB** | **ALUM2Reg** | **RegWre** | **InsMemRW** | **DataMemRW** | **ExtSel** | **PCSrc** | **RegOut** | **ALUOp[2..0]** |
| add | x | 1 | 0 |  | 1 |  |  |  |  |  | 000 |
| addi | x | 1 | 1 |  | 1 |  |  |  |  |  | 000 |
| sub | x | 1 | 0 |  | 1 |  |  |  |  |  | 001 |
| ori | x | 1 | 1 |  | 1 |  |  |  |  |  | 011 |
| and | x | 1 | 0 |  | 1 |  |  |  |  |  | 100 |
| or | x | 1 | 0 |  | 1 |  |  |  |  |  | 011 |
| move | x | 1 | 0 |  | 1 |  |  |  |  |  | 000 |
| sw | x | 1 | 1 |  | 0 |  |  |  |  |  | 000 |
| lw | x | 1 | 1 |  | 1 |  |  |  |  |  | 000 |
| beq | 0 | 1 | 0 |  | 0 |  |  |  |  |  | 001 |
| 1 | 1 | 0 |  | 0 |  |  |  | 1 |  | 001 |
| halt | x | 0 | x |  | x |  |  |  |  |  | xxx |

说明：上表是根据“实验一单周期CPU设计”的有关要求得出来的，未完成的部分留给学生。

以上指令相应的变量假定是这样表示的，指令助记符前加“i\_”，如“add”，则变量为“i\_add”，以此类推。指令代码32为长度，最高6位为操作码，唯一标识一条汇编指令，如规定操作码op=“000000”为加法指令，即助记符add。如果从指令存储器中读出来的指令代码，其操作码为“000000”，则规定在控制器中i\_add=1，否则i\_add=0。以此类推。

以上‘z’表示标志位，运算结果为z=1，否则z=0，以下用zero表示。

**逻辑表达式：**

**ALUSrcB=i\_addi | i\_ori | i\_sw | i\_lw**

**……**

**RegWre=i\_add | i\_addi | i\_sub | i\_ori | i\_and | i\_or | i\_move | i\_lw**

**……**

**ALUOp[2]=i\_and**

**ALUOp[1]=i\_ori | i\_or**

**ALUOp[0]=i\_sub | i\_ori | i\_or | i\_beq**

**……**

**当然，也可以这样考虑控制单元的子模块内容：**

**module ControlUnit(**

**opcode,**

**Zero,**

**RegWre,**

PCWre**,**

**ALUSrcB,**

**ALUOp,**

**ALUM2Reg,**

**RegOut,**

**DataMemRW,**

**PCSrc,**

**ExtSel**

**);**

**input [5:0] opcode,zero; //指令操作码**

**output reg RegWre, PCWre, ALUSrcB, ALUM2Reg, RegWre, DataMemRW, PCSrc, ExtSel;**

**output reg [2:0] ALUOp;**

**//若opcode(指令操作码)有变化，都会触发以下部分产生新的控制信号**

**// 如果beq指令时，PCSrc=zero。**

**always@( opcode) begin**

**case( opcode )**

**// add, R-format**

**6'b000000:**

**Begin //以下都是控制单元产生的控制信号**

**RegWre = 1;**

**PCWre = 1; // PC写使能**

**ALUSrcB = 0; // ALU B口数据选择控制**

**ALUOp = 000; // ALU功能码**

**ALUM2Reg = 0; // ALU或存储器的数据送往寄存器**

**RegWre = 1; // 寄存器写使能**

**DataMemRW = 0; // 1为写，0为读**

**PCSrc = 0; // PC转移控制**

**ExtSel = 0; // immediate扩展控制**

**end**

**// sub, R-format**

**6'b000001:**

**begin**

**RegWre = 1;**

**PCWre = 1;**

**ALUSrcB = 0;**

**ALUOp = 001;**

**ALUM2Reg = 0;**

**RegWre = 1;**

**DataMemRW = 0;**

**PCSrc = 0;**

**ExtSel = 0;**

**end**

**……**

**end**

**endmodule**