

SHANGHAI JIAOTONG UNIVERSITY



计算机系统结构实验报告 - Lab3

姓名：刘欣鹏

学号：516030910259

完成时间：2018/3/7

目录

[1. 概述 3](#_Toc19913)

[1.1 实验名称 3](#_Toc31792)

[1.2 实验目的 3](#_Toc31334)

[1.3实验范围 3](#_Toc2936)

[2. 实验描述 3](#_Toc32084)

[2.1 主控制单元模块CTR 3](#_Toc8184)

[2.1.1模块描述 3](#_Toc22807)

[2.1.2 CTR模块代码 5](#_Toc28039)

[2.1.3 仿真测试代码 7](#_Toc18434)

[2.1.4 仿真波形 7](#_Toc11166)

[2.1.5 实验结论 7](#_Toc14419)

[2.2 ALU单元模块ALUCtr 8](#_Toc31108)

[2.2.1 模块描述 8](#_Toc5683)

[2.2.2 ALUCTR单元模块代码 9](#_Toc7380)

[2.2.3 仿真测试代码 9](#_Toc14999)

[2.2.4 仿真波形 10](#_Toc13645)

[2.2.5 实验结论 10](#_Toc20059)

[2.3 ALU 10](#_Toc6158)

[2.3.1 模块描述 10](#_Toc11026)

[2.3.2 ALU模块代码 11](#_Toc21919)

[2.3.3 仿真测试代码 12](#_Toc20310)

[2.3.4 仿真波形 13](#_Toc18824)

[2.3.5 实验结论 13](#_Toc27297)

[3. 实验心得 13](#_Toc18232)

# 1. 概述

## 1.1 实验名称

简单的类MIPS单周期处理器实现——控制器、ALU

## 1.2 实验目的

理解CPU控制器、ALU的原理

## 1.3实验范围

本次实验将覆盖以下范围:

(1) ISE的使用

(2) Spartan-3E实验板的使用

(3)使用Verilog HDL进行逻辑设计

(4) CPU控制器的实现

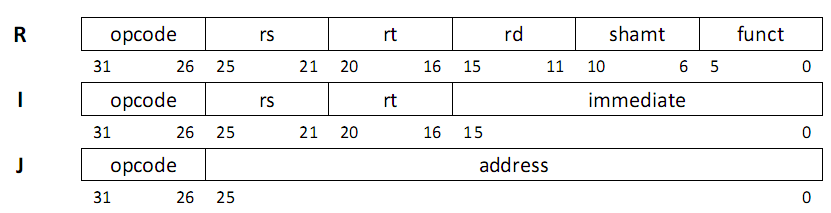
(5) ALU的实现

# 2. 实验描述

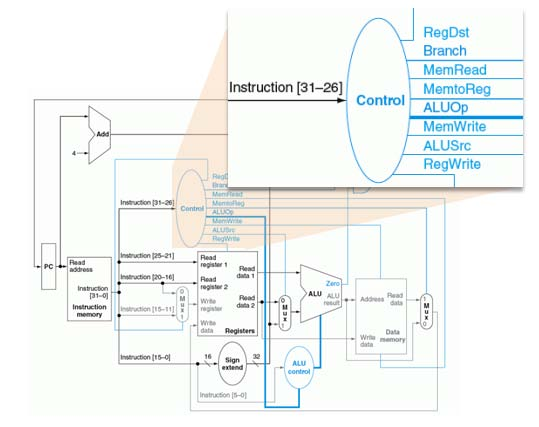
## 2.1 主控制单元模块CTR

### 2.1.1模块描述

主控制单元输入为指令的opCode字段，即操作码。操作码经过主控制单元的译码，为ALUCtr，Data Memory，Registers，Muxs等部件输出正确的控制信号。



Mips基本指令格式



主控制模块（控制模块的IO定义）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Input or output** | **Signal name** | **R-format**  **(add, sub, and, or , slt)** | **lw** | **sw** | **beq** | **jump** |
| Inputs | Op5 | 0 | 1 | 1 | 0 | 0 |
| Op4 | 0 | 0 | 0 | 0 | 0 |
| Op3 | 0 | 0 | 1 | 0 | 0 |
| Op2 | 0 | 0 | 0 | 1 | 0 |
| Op1 | 0 | 1 | 1 | 0 | 1 |
| Op0 | 0 | 1 | 1 | 0 | 0 |
| Outputs | RegDst | 1 | 0 | X | X | X |
| ALUSrc | 0 | 1 | 1 | 0 | X |
| MemtoReg | 0 | 1 | X | X | X |
| RegWrite | 1 | 1 | 0 | 0 | 0 |
| MemRead | 0 | 1 | 0 | 0 | 0 |
| MemWrite | 0 | 0 | 1 | 0 | 0 |
| Branch | 0 | 0 | 0 | 1 | 0 |
| ALUOp | 10 | 00 | 00 | 01 | XX |
| Jump | 0 | 0 | 0 | 0 | 1 |

主控制模块真值表（opCode与控制输出的编码关系）

### 2.1.2 CTR模块代码

**module ctr(**

**input [5:0] opCode,**

**output regDst,**

**output aluSrc,**

**output memToReg,**

**output regWrite,**

**output memRead,**

**output memWrite,**

**output branch,**

**output [1:0] aluOp,**

**output jump**

**);**

**reg regDst;**

**reg aluSrc;**

**reg memToReg;**

**reg regWrite;**

**reg memRead;**

**reg memWrite;**

**reg branch;**

**reg [1:0] aluOp;**

**reg jump;**

**always @ (opCode)**

**begin**

**case (opCode)**

**6'b000010:**

**begin**

**regDst=0;**

**aluSrc=0;**

**memToReg=0;**

**regWrite=0;**

**memRead=0;**

**memWrite=0;**

**branch=0;**

**aluOp=2'b00;**

**jump=1;**

**end**

**6'b000000:**

**begin**

**regDst=1;**

**aluSrc=0;**

**memToReg=0;**

**regWrite=1;**

**memRead=0;**

**memWrite=0;**

**branch=0;**

**aluOp=2'b10;**

**jump=0;**

**end**

**6'b100011:**

**begin**

**regDst=0;**

**aluSrc=1;**

**memToReg=1;**

**regWrite=1;**

**memRead=1;**

**memWrite=0;**

**branch=0;**

**aluOp=2'b00;**

**jump=0;**

**end**

**6'b101011:**

**begin**

**regDst=0;**

**aluSrc=1;**

**memToReg=0;**

**regWrite=0;**

**memRead=0;**

**memWrite=1;**

**branch=0;**

**aluOp=2'b00;**

**jump=0;**

**end**

**6'b000100:**

**begin**

**regDst=0;**

**aluSrc=0;**

**memToReg=0;**

**regWrite=0;**

**memRead=0;**

**memWrite=0;**

**branch=1;**

**aluOp=2'b01;**

**jump=0;**

**end**

**default:**

**begin**

**regDst=0;**

**aluSrc=0;**

**memToReg=0;**

**regWrite=0;**

**memRead=0;**

**memWrite=0;**

**branch=0;**

**aluOp=2'b00;**

**jump=0;**

**end**

**endcase**

**end**

**endmodule**

### 2.1.3 仿真测试代码

仿真代码主要部分如下，完整代码见test\_for\_Ctr.v。

**initial begin**

**// Initialize Inputs**

**opCode = 0;**

**// Wait 100 ns for global reset to finish**

**#100;**

**#100 opCode=6'b000010; //jump**

**#100 opCode=6'b000000; //R-format**

**#100 opCode=6'b100011; //lw**

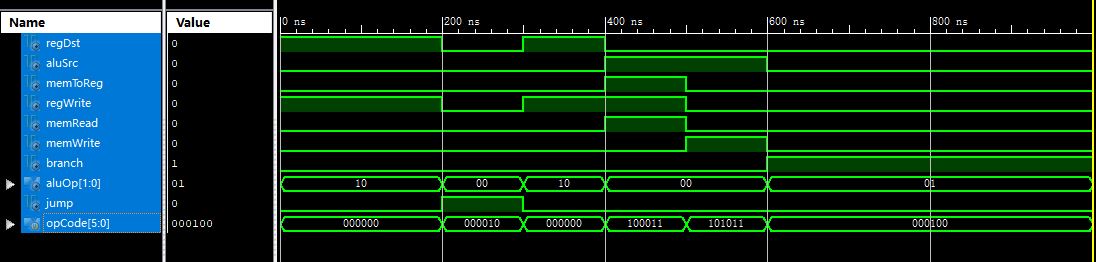
**#100 opCode=6'b101011; //sw**

**#100 opCode=6'b000100; //beq**

**// Add stimulus here**

**end**

### 2.1.4 仿真波形



CTR模块仿真波形

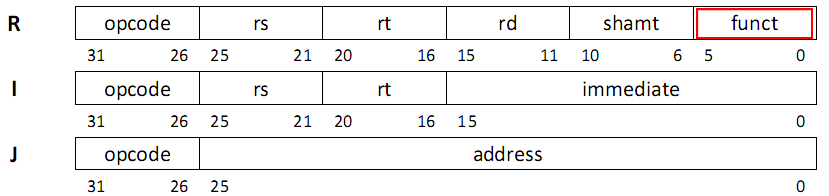
### 2.1.5 实验结论

仿真波形与CTR模块真值表完全对应，能够根据不同的opCode输出正确的控制信号，主控制模块实验成功。

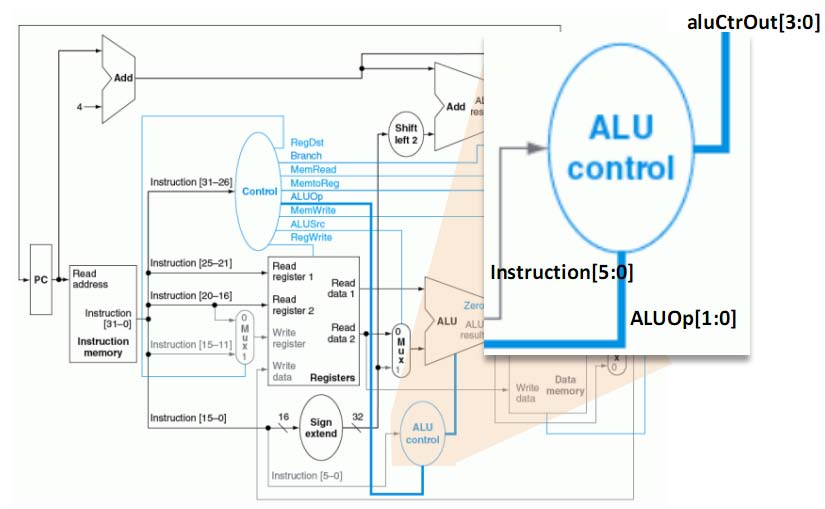
## 2.2 ALU单元模块ALUCtr

### 2.2.1 模块描述

ALUCtr根据主控制器的ALUOp来判断指令类型，根据指令的后六位区分R型指令，综合这两种输入，控制ALU做正确的操作。

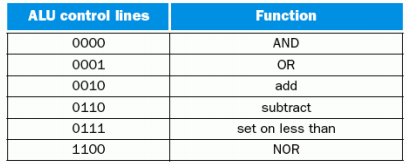


指令的后六位用于区分R型指令

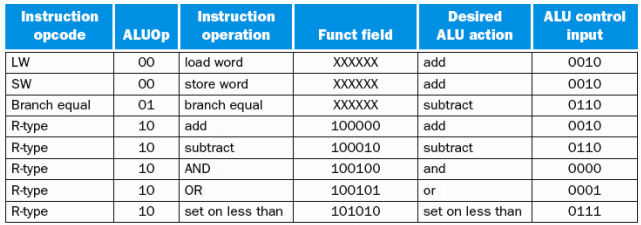


ALU控制模块的IO定义

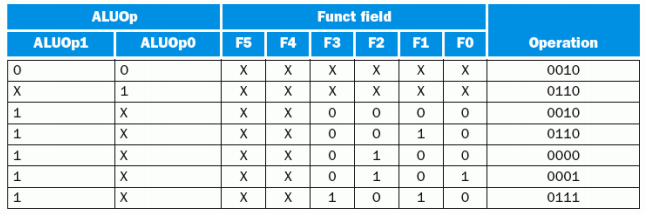
我们约定ALUCtrOut[3:0]与ALU操作的对应关系如下：



ALUCtrOut和ALU操作的对应关系



根据ALUOp控制位和R型指令的不同功能码来设ALU控制位



输入输出真值表

### 2.2.2 ALUCTR单元模块代码

**module AluCtr(**

**input [1:0] aluOp,**

**input [5:0] funct,**

**output [3:0] aluCtr**

**);**

**reg [3:0] aluCtr;**

**always @ (aluOp or funct)**

**casex ({aluOp,funct})**

**8'b00xxxxxx: aluCtr=4'b0010;**

**8'bx1xxxxxx: aluCtr=4'b0110;**

**8'b1xxx0000: aluCtr=4'b0010;**

**8'b1xxx0010: aluCtr=4'b0110;**

**8'b1xxx0100: aluCtr=4'b0000;**

**8'b1xxx0101: aluCtr=4'b0001;**

**8'b1xxx1010: aluCtr=4'b0111;**

**default: aluCtr=4'b1111;**

**endcase**

**endmodule**

### 2.2.3 仿真测试代码

仿真代码主要部分如下，完整代码见test\_for\_AluCtr.v。

**initial begin**

**// Initialize Inputs**

**aluOp = 0;**

**funct = 0;**

**// Wait 100 ns for global reset to finish**

**#100;**

**#100 aluOp=2'b00;funct=6'b000000;**

**#100 aluOp=2'b01;funct=6'b000000;**

**#100 aluOp=2'b10;funct=6'b100000;**

**#100 aluOp=2'b10;funct=6'b100010;**

**#100 aluOp=2'b10;funct=6'b100100;**

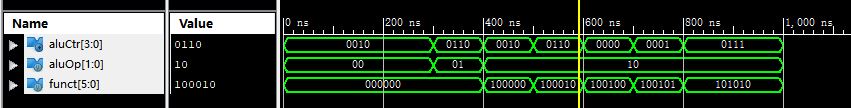
**#100 aluOp=2'b10;funct=6'b100101;**

**#100 aluOp=2'b10;funct=6'b101010;**

**// Add stimulus here**

**end**

### 2.2.4 仿真波形



ALUCTR模块仿真波形

### 2.2.5 实验结论

仿真波形与ALUCTR模块真值表完全对应，能够根据不同的ALUOp及Funct的输入，输出正确的ALU控制信号，ALU单元模块实验成功。

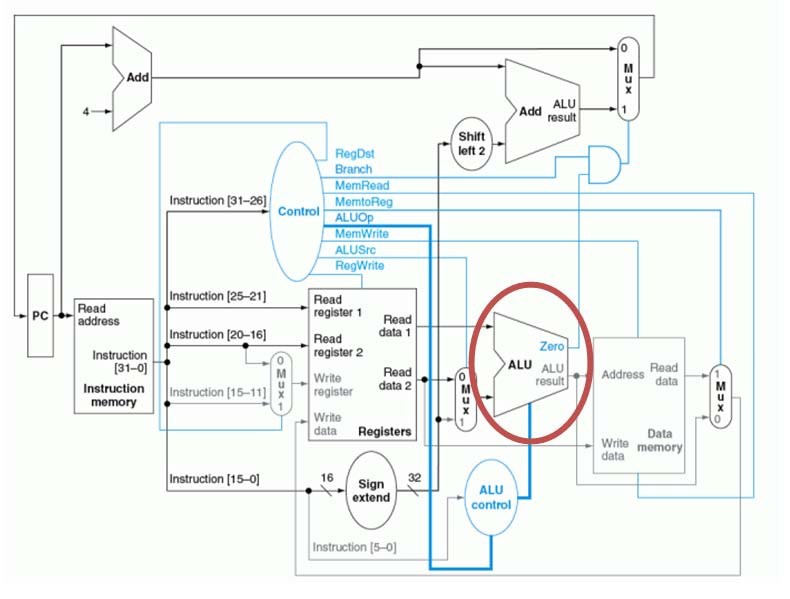
## 2.3 ALU

### 2.3.1 模块描述

根据ALUCtr，对两个输入做对应的操作，由ALURes输出结果。若结果为0，zero输出置1。

输入：input1 (32 bits), input2 (32 bits), aluCtr (4 bits)

输出：zero (1 bit), ALURes (32 bits)



ALU模块的IO定义

### 2.3.2 ALU模块代码

**module alu(**

**input [31:0] input1,**

**input [31:0] input2,**

**input [3:0] aluCtr,**

**output zero,**

**output [31:0] aluRes**

**);**

**reg zero;**

**reg [31:0] aluRes;**

**always @ (input1 or input2 or aluCtr)**

**begin**

**if (aluCtr == 4'b0010) //add**

**begin**

**aluRes=input1+input2;**

**if (aluRes==0)**

**zero=1;**

**else**

**zero=0;**

**end**

**else if (aluCtr == 4'b0110) //sub**

**begin**

**aluRes=input1-input2;**

**if (aluRes==0)**

**zero=1;**

**else**

**zero=0;**

**end**

**else if (aluCtr == 4'b0000)**

**begin**

**aluRes=input1 & input2;**

**if (aluRes == 0)**

**zero=1;**

**else**

**zero=0;**

**end**

**else if (aluCtr == 4'b0001)**

**begin**

**aluRes=input1 | input2;**

**if (aluRes == 0)**

**zero=1;**

**else**

**zero=0;**

**end**

**else if (aluCtr == 4'b0111)**

**begin**

**if (input1 < input2)**

**begin**

**aluRes=1;zero=0;**

**end**

**else begin**

**aluRes=0;zero=1;**

**end**

**end**

**else if (aluCtr == 4'b1100)**

**begin**

**aluRes=~(input1 | input2);**

**if (aluRes == 0)**

**zero=1;**

**else**

**zero=0;**

**end**

**end**

**endmodule**

### 2.3.3 仿真测试代码

仿真代码主要部分如下，完整代码见test\_for\_Alu.v。

**initial begin**

**// Initialize Inputs**

**input1 = 255;**

**input2 = 170;**

**aluCtr = 0;**

**// Wait 100 ns for global reset to finish**

**#100;**

**#100 aluCtr=4'b0000;**

**#100 aluCtr=4'b0001;**

**#100 aluCtr=4'b0010;**

**#100 aluCtr=4'b0110;**

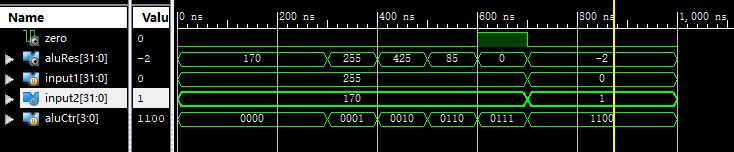
**#100 aluCtr=4'b0111;**

**#100 input1=0;input2=1;aluCtr=4'b1100;**

**// Add stimulus here**

**end**

### 2.3.4 仿真波形



ALU模块仿真波形

### 2.3.5 实验结论

仿真波形与ALU模块真值表完全对应，能够根据不同的ALUCtr，对input1和input2做出正确的运算，输出到zero和ALURes。ALU模块实验成功。

# 3. 实验心得

本次实验总体来说难度其实不是很大，但是由于还没有学习过相关的理论知识，实验的过程中会遇到一些让人感到困惑的地方。好在实验指导书对此实验的说明细致详尽，老师也为我耐心地解答了疑问，最终实验取得了成功。

这次实验使我初步了解了verilog的一些基本语法，为之后的实验做了一定的准备。

这次实验我做得相对而言比较快，这主要归功于实验指导书的细致说明和指导老师的耐心讲解。在此，感谢老师和同学们的帮助。