Lab03 实验报告

王正 518021910079

一、 实验名称

简单的类 MIPS 单周期处理器部件的实现–控制器，ALU

二、 实验目的

1. 理解CPU 控制器，ALU 的原理
2. 控制器Ctr 的实现
3. 运算单元控制器ALUCtr 的实现
4. ALU的实现
5. 使用功能仿真

三、 实验原理

1. 控制器 Ctr

主控制单元为一个译码器，其接受指令的 [31:26] 位字段（opCode）作为输入，给 ALU 控制器、数据内存、寄存器和数据选择器输出正确的控制信号。

2. 运算单元控制器 ALUCtr

ALU 控制单元模块（ALUCtr）根据主控制器的 ALUOp 判断指令类型，从而向 ALU 输出正确的运算控制信号。R类型指令是根据指令的低 6 位（Funct）来的判断的，ALUCtr 会综合 ALUOp 和 Funct 来进行译码。

3. 算逻单元 ALU

根据 ALUCtr 的控制信号，对两个操作数进行某个逻辑或算术运算，将结果输出到 ALURes 中。如果结果为零，将Zero 设为真。

四、 功能实现

1. 控制器 Ctr

利用 case 语句来实现:

```
module Ctr(
    input [5:0] OpCode,
    output reg regDst,
    output reg aluSrc,
    output reg memToReg,
    output reg regWrite,
    output reg memRead,
    output reg memWrite,
    output reg branch,
    output reg [1:0] aluOp,
    output reg jump
    );

  always @(OpCode)
  begin
      case(OpCode)
          6'b000000://R
          begin
              regDst=1;
              aluSrc=0;
              memToReg=0;
              regWrite=1;
              memRead=0;
              memWrite=0;
              branch=0;
              aluOp=2'b10;
              jump=0;
          end


      6'b100011://lw
      begin
          regDst=0;
          aluSrc=1;
          memToReg=1;
          regWrite=1;
          memRead=1;
          memWrite=0;
          branch=0;
          aluOp=2'b00;
          jump=0;
      end

      6'b101011://sw
      begin
         regDst=0;
          aluSrc=1;
          memToReg=0;
          regWrite=0;
          memRead=0;
          memWrite=1;
          branch=0;
          aluOp=2'b00;
          jump=0;
      end
```

```
            6'b000100://beq
            begin
               regDst=0;
                aluSrc=0;
                memToReg=0;
                regWrite=0;
                memRead=0;
                memWrite=0;
                branch=1;
                aluOp=2'b01;
                jump=0;
            end

            6'b000010://J
            begin
               regDst=0;
                 aluSrc=0;
                memToReg=0;
                regWrite=0;
                memRead=0;
                memWrite=0;
                branch=0;
                aluOp=2'b00;
                jump=1;
            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
```

2. 运算单元控制器 ALUCtr

与 Ctr 的实现类似，同样适用 case 语句:

```
module ALUCtr(
    input [1:0] ALUOp,
    input [5:0] Funct,
    output reg [3:0] ALUCtrOut
    );

    always @(ALUOp, Funct)
    begin
        casex({ALUOp,Funct})
            8'b00000000:ALUCtrOut=4'b0010;
            8'b01000000:ALUCtrOut=4'b0110;
            8'b10000000:ALUCtrOut=4'b0010;
            8'b10000010:ALUCtrOut=4'b0110;
            8'b10000100:ALUCtrOut=4'b0000;
            8'b10000101:ALUCtrOut=4'b0001;
            8'b10001010:ALUCtrOut=4'b0111;
        endcase
    end
endmodule
```

3. 算逻单元 ALU

根据 ALUCtr 信号，对操作数进行相应运算即可。

```
module ALU(input1, input2, aluCtr, zero, aluRes );
    input [31:0] input1;
    input [31:0] input2;
    input [3:0] aluCtr;
    output reg zero;
    output reg [31:0] aluRes;

    always @ (input1 or input2 or aluCtr)
    begin
       case(aluCtr)
    4'b0000:
    begin
    aluRes=input1 & input2;
    end
    4'b0001:
    begin
    aluRes=input1 | input2;
    end
    4'b0010:
    begin
    aluRes=input1+input2;
    end
    4'b0110:
    begin
    aluRes=input1-input2;
    end

    4'b0111:
    begin
        if (input1<input2)
            aluRes=1;
        else
            aluRes=0;
    end
    4'b1100:
    begin
    aluRes=~(input1|input2);
    end
    default:
    begin
    aluRes=0;
    end
 endcase
  if (aluRes==0)
        zero=1;
    else
        zero=0;
    end

endmodule
```

五、 激励文件

1. Ctr_tb:

```
module Ctr_tb();
    reg [5:0] OpCode;
    wire regDst;
    wire aluSrc;
    wire memToReg;
    wire regWrite;
    wire memRead;
    wire memWrite;
    wire branch;
    wire [1:0] aluOp;
    wire jump;

    Ctr u0 (
        .OpCode(OpCode),
        .regDst(regDst),
        .aluSrc(aluSrc),
        .memToReg(memToReg),
        .regWrite(regWrite),
        .memRead(memRead),
        .memWrite(memWrite),
        .branch( branch),
        . aluOp( aluOp),
        .jump(jump)
        );


     initial begin
         OpCode=0;
         #100;

         #100 OpCode=6'b000000;
         #100 OpCode=6'b100011;
         #100 OpCode=6'b101011;
         #100 OpCode=6'b000100;
         #100 OpCode=6'b000010;
         #100 OpCode=6'b010101;
         end
endmodule
```

2. ALUCtr_tb

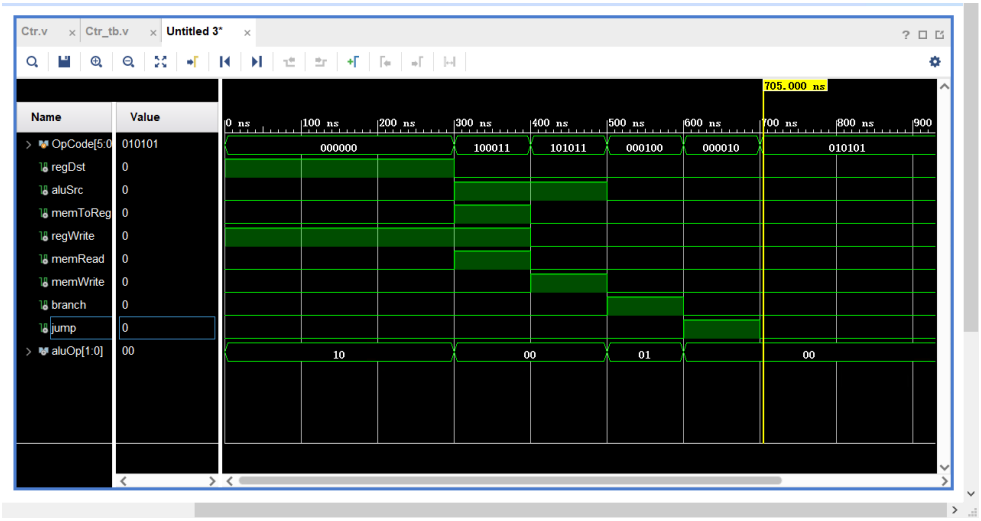
```
module ALUCtr_tb();
    reg [1:0] ALUOp;
    reg [5:0] Funct;
    wire [3:0] ALUCtrOut;

    ALUCtr u0(
       .ALUOp(ALUOp),
       .Funct(Funct),
       .ALUCtrOut(ALUCtrOut)
       );

    initial begin
       ALUOp=0;
       Funct=0;
       #100;

      #100 ALUOp=2'b00;Funct=6'b000000;
      #100 ALUOp=2'b01;Funct=6'b000000;
      #100 ALUOp=2'b10;Funct=6'b000000;
      #100 ALUOp=2'b10;Funct=6'b000010;
      #100 ALUOp=2'b10;Funct=6'b000100;
      #100 ALUOp=2'b10;Funct=6'b000101;
      #100 ALUOp=2'b10;Funct=6'b001010;

       end
endmodule
```

3. ALU_tb

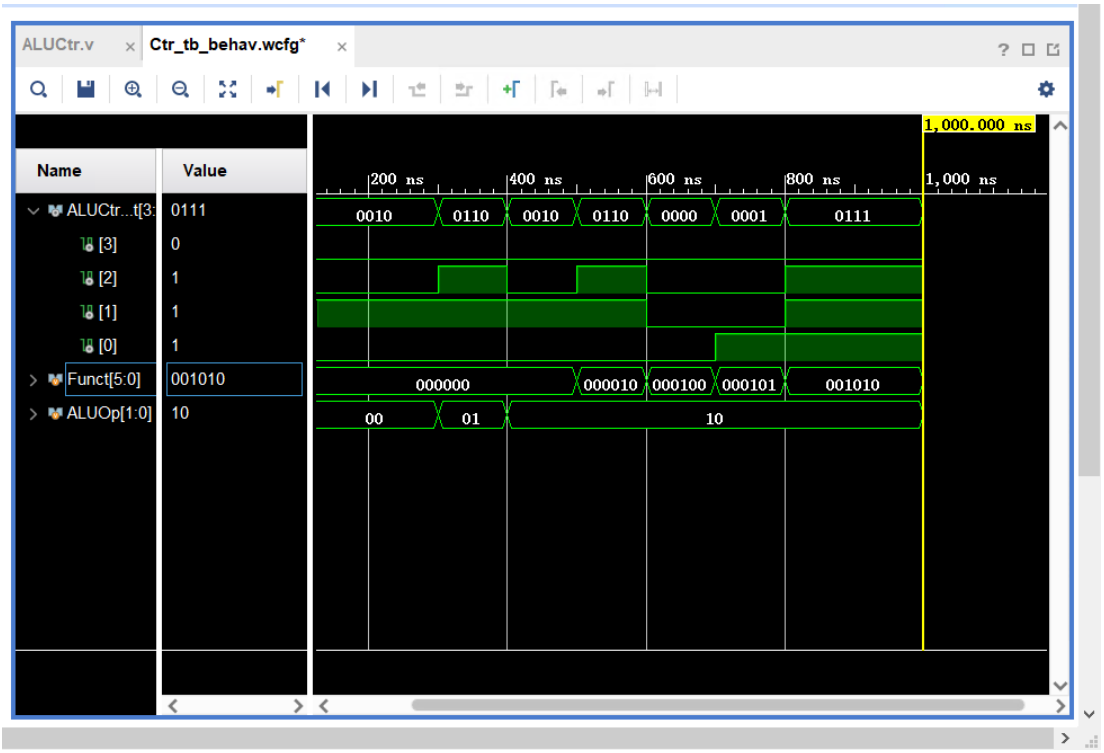
```
module ALU_tb( );
    reg [31:0] input1;
    reg [31:0] input2;
    reg [3:0] aluCtr;
    wire zero;
    wire [31:0] aluRes;

    ALU uo (
        .input1(input1),
        .input2(input2),
        .aluCtr(aluCtr),
        .zero(zero),
        .aluRes(aluRes) );

   initial begin
      aluCtr=0;
      input1=0;
      input2=0;

      #100 input1=15;input2=10;aluCtr=4'b0000;
      #100 aluCtr=4'b0001;
      #100 aluCtr=4'b0010;
      #100 aluCtr=4'b0110;
      #100 input1=10;input2=15;aluCtr=4'b0110;
      #100 input1=15;input2=10;aluCtr=4'b0111;
      #100 input1=10;input2=15;aluCtr=4'b0111;
      #100 input1=1;input2=1;aluCtr=4'b1100;
      #100 input1=16;input2=1;aluCtr=4'b1100;
      end
endmodule
```

六、 结果展示

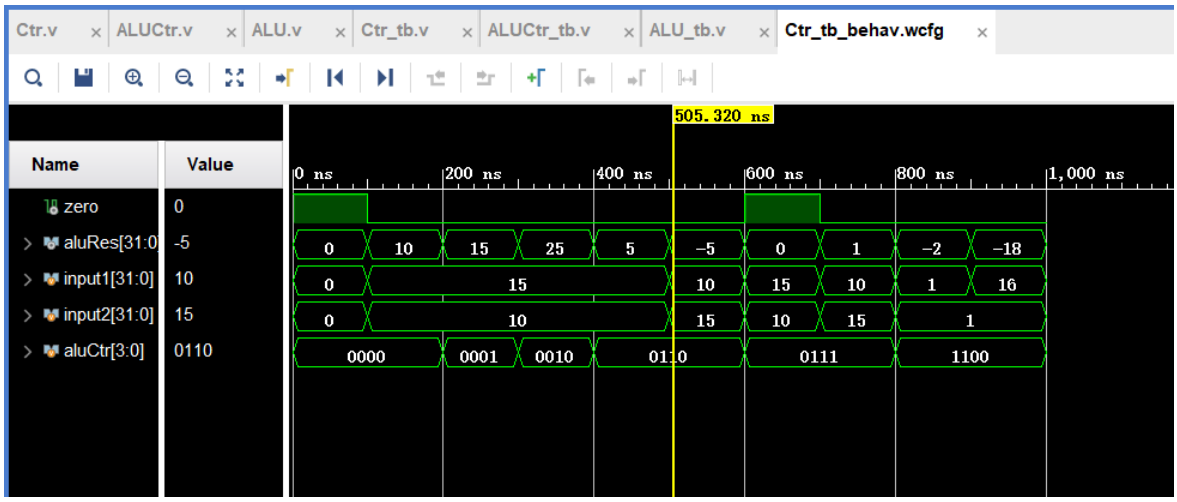
1. Ctr 仿真波形



2. ALUCtr 仿真波形



3. ALU 仿真



由波形可知，仿真结果满足预期设计。

七、 心得体会

本次实验实现了简单 MIPS 处理器中的重要的运算单元和控制单元。这次实验的难度较前面两次有所增加，指导书也没有给出完整的代码，需要自己根据 ALU 的工作原理进行思考来补全代码。通过这次实验，我对 CPU 控制器，ALU 的工作原理有了更进一步的认识。